CIU TOUR: Reconocimiento de edificios y monumentos mediante aprendizaje automático

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DEL SOFTWARE

CURSO 2018–2019

Realizado por

Arturo Marino Quintana
Stefano Mazzuka Cassani

Director

Rubén Fuentes

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática
Universidad Complutense de Madrid
Madrid. Junio de 2019

Agradecimientos

A toda mi familia y amigos, gracias a quienes soy quien soy y hacia quienes sólo puedo expresar mi más sincero agradecimiento. Gracias a mi mujer, Nazareth, sin cuyo apoyo no habría podido concluir esta etapa, ni comenzar la siguiente empezando con el nacimiento de mi primer hijo pocos días después de la entrega de esta memoria.

Arturo Marino Quintana

En primer lugar, a los profesores que me han ayudado a crecer como profesional.
A mis familiares y amigos, que me animaron a seguir hasta el final.
A los más jóvenes con Trastorno por Déficit de Atención o Dislexia quiero animarlos a continuar con sus estudios y alcanzar sus metas personales.

Stefano Mazzuka Cassani

Índice

Índice de figuras

Índice de tablas

Resumen

El sector turístico busca permanentemente nuevas propuestas para mejorar las experiencias de sus clientes. Fruto del uso de nuevas tecnologías principalmente a través de dispositivos móviles, las personas desean en la actualidad obtener una mayor información de su entorno con el menor esfuerzo posible. En el turismo, esta tendencia se plasma, entre otras propuestas, en mejorar las experiencias de los usuarios para que sean cada vez más inmersivas y cómodas. En este contexto surge este proyecto.

El objetivo principal de este proyecto ha sido el diseño e implementación de una aplicación móvil para visitas turísticas que permite al usuario obtener información sobre edificios y monumentos del entorno. Para ello, se ha realizado una captura de los requisitos necesarios para el desarrollo de la aplicación mediante un análisis de la competencia, extrayéndose requisitos funcionales a nivel de usuario y administración, y también de usabilidad. También se han estudiado las herramientas de aprendizaje automático disponibles para la clasificación de imágenes. Ello ha permitido el uso e implementación de una red neuronal para la clasificación de imágenes.

El resultado es una aplicación capaz de obtener de manera sencilla una imagen del entorno, clasificarla en una categoría (que corresponde a un edificio o monumento) y mostrar información relacionada con ésta. De este modo, el usuario puede realizar su propio tour virtual usando la información del proveedor.

La aplicación se ha evaluado desde el punto de vista de la clasificación y reconocimiento de imágenes y de la experiencia del usuario. Se ha concluido que la aplicación ha satisfecho en general a los usuarios en cuanto a la funcionalidad, utilidad, comodidad e innovación, a pesar de posibles correcciones o mejoras que puedan realizarse. Cabe destacar que, tras analizar el proceso de clasificación, se detectó la necesidad de realizar una optimización de dicho proceso mediante la ampliación del número y variedad del conjunto de imágenes usadas para el entrenamiento de la red neuronal y el uso de herramientas de optimización ofrecidas por TensorFlow.

Como mejoras futuras de la aplicación, se plantea la optimización del proceso de clasificación, la implementación de realidad virtual y/o realidad aumentada, y mejoras en la interfaz de usuario.

Palabras clave: aprendizaje automático, reconocimiento de imágenes, clasificación, TensorFlow, turismo.

Abstract

The tourism sector is constantly looking for new proposals to improve the experiences of its customers. As a result of the use of new technologies, mainly through mobile devices, people currently want to obtain more information about their environment with the least possible effort. In tourism, this trend is reflected, among other proposals, in improving the experiences of users to become increasingly immersive and comfortable. It is in this context that this project arises.

The main objective of this project has been the design and implementation of a mobile application for tourists that allows the user to obtain information on buildings and monuments in the environment. To this end, a capture of the necessary requirements for the development of the application has been carried out through an analysis of the competition, extracting functional requirements at user and administration level, as well as usability. The automatic learning tools available for image classification have also been studied. This has allowed the use and implementation of a neural network for image classification.

The result is an application capable of easily obtaining an image of the environment, classifying it into a category (corresponding to a building or monument) and displaying information related to it. In this way, the user can make his own virtual tour using the information of the provider.

The application has been evaluated from the point of view of the classification and recognition of images and the user experience. It has been concluded that the application has generally satisfied users in terms of functionality, usefulness, comfort and innovation, despite possible corrections or improvements that may be made. It should be noted that after analyzing the classification process, it was detected the need to perform an optimization of this process by increasing the number and variety of images used for training the neural network and the use of optimization tools offered by TensorFlow.

Future improvements to the application include the optimization of the classification process, the implementation of virtual reality and/or augmented reality, and improvements to the user interface.

Keywords: automatic learning, image recognition, classification, TensorFlow, tourism.

1. Introducción

Este proyecto nace con la idea de permitir a las personas obtener información de su entorno con el menor esfuerzo posible, mejorando la experiencia turística del usuario. Para ello, no basta únicamente con listar montañas de información, obligando al usuario a buscar un contenido concreto de un elemento, sino mostrar al usuario de forma sencilla y cómoda información de lo que está viendo, a través del reconocimiento y clasificación de imágenes desde su propio teléfono móvil.

En esta sección se especificarán los objetivos proyectados para la implementación de este trabajo (sección 1.1) y el plan de trabajo seguido durante dicho proceso (sección 1.2). Además, se expone una introducción al aprendizaje automático (sección 1.3) como base para entender la tecnología usada en el desarrollo de la aplicación para la clasificación de imágenes.

1.1. Objetivos

El **objetivo general** de este proyecto es el diseño e implementación de una aplicación móvil para visitas turísticas. La aplicación permitirá al usuario obtener información sobre algunos edificios y monumentos de su entorno a partir de la captura de imágenes. El reconocimiento de imágenes se basará en técnicas de aprendizaje automático.

Este objetivo general se puede dividir en los siguientes objetivos específicos:

1- **Análisis de la competencia.** Busca recopilar características de distintas aplicaciones relacionadas con el sector turístico y/o con el reconocimiento de imágenes. Se trata de obtener los criterios mínimos que cubran la demanda de los usuarios. Además, este análisis nos permite esclarecer las fortalezas y debilidades respecto a la usabilidad y comodidad de cada aplicación para incorporarlos o evitarlos en la aplicación desarrollada.

2- **Análisis sobre las tecnologías de aprendizaje automático actuales y elección de la tecnología más adecuada al proyecto.** Para ello se tiene en cuenta su uso, su tipo de licencia, su precio, la comunidad de desarrolladores que la apoya y las capacidades de reconocimiento de imágenes.

3- **Generación del sistema de aprendizaje para la detección de objetos (edificios y monumentos)** en base a la tecnología de aprendizaje automática seleccionada. Este paso incluye además la validación del modelo generado.

4- **Diseño e implementación del sistema software.** Se aplican conocimientos de arquitectura de software aprendidos durante el trascurso del grado. También incluye la realización de baterías de pruebas para la detección de posibles errores en el diseño y código fuente.

5- **Evaluación de la aplicación.** Incluye dos evaluaciones diferentes. Por un lado, una evaluación por parte de usuarios reales, centrada en la funcionalidad, utilidad, comodidad, innovación, satisfacción y posibles mejoras futuras. Por otro lado, una

batería de pruebas para comprobar la efectividad de las técnicas de aprendizaje automático en función del entrenamiento recibido en cada objeto.

A partir de la información obtenida al resolver los objetivos anteriores, se ha elegido para el desarrollo la plataforma Android y para el aprendizaje TensorFlow. El caso de estudio usado corresponde a visitas en la Ciudad Universitaria (Madrid, España).

1.2. Plan de trabajo

La metodología de trabajo se establece en función de las características del proyecto a desarrollar. Entre ellas hay que tener en cuenta aspectos del sistema a construir, tales como la dificultad de predecir qué requisitos software persistirán durante todo el proceso de desarrollo, al haber una exploración de los mismos con los usuarios finales. También la incertidumbre tecnológica, al no tener conocimientos sobre la tecnología específica a aplicar y la dificultad de integración de múltiples elementos. Además, hay que tener en cuenta las restricciones relacionadas con el equipo de trabajo, tales como el número de desarrolladores, su experiencia y conocimientos, su disponibilidad, y los eventos y posibles imprevistos de cada miembro del grupo a lo largo del curso.

Dada la incertidumbre sobre algunos aspectos y la flexibilidad al abordar el proyecto, se establece la idoneidad de adoptar una metodología ágil. En este caso se opta por la metodología Scrum (Sutherland, 1997) (ver Sección 1.2.1), adaptando la complejidad del método a las circunstancias del equipo de desarrollo (ver Sección 1.2.2).

1.2.1. Scrum

Scrum es un modelo de proceso desarrollado por Jeff Sutherland en 1997, y revisado posteriormente por Schwaber y Beedle (2002). Está basado en un estudio sobre los procesos de desarrollo utilizados en Japón y Estados Unidos (Takeuchi & Nonaka, 1986). Scrum es "un marco de trabajo para desarrollar, entregar y mantener productos complejos" (Schwaber & Sutherland, 2017, pp. 3) basado en iteraciones cortas y fijas (sprints). Estas suelen durar de dos semanas a un mes, tiempo durante el cual se realiza un desarrollo incremental sobre el producto, repitiéndose un proceso de trabajo similar cada vez, para proporcionar un resultado completo y funcional. En la siguiente iteración, el desarrollo se realizará sobre el resultado de la iteración anterior (*Product Backlog*) mejorando el desarrollo completado anteriormente o añadiendo nuevos objetivos o requisitos.

Este proceso iterativo e incremental mejora la productividad del proyecto, dado que el equipo de desarrollo trabaja de forma más eficiente al tener objetivos a corto plazo y que la precisión de las estimaciones aumenta al ser un proceso corto. Además, permite conocer el proceso real del proyecto.

Al comienzo de cada *sprint*, se realiza una planificación (*Sprint Plannig*) donde se planifica:

- Qué se desarrollará en la siguiente iteración (*Sprint Goals*). Para ello, se seleccionan los objetivos/requisitos más prioritarios en función del valor de negocio a añadir respecto al esfuerzo y riesgo, para optimizar la productividad y calidad del producto.

- Cómo se desarrollará. Para ello se define una lista con las tareas necesarias para cumplir con el objetivo (*Sprint Backlog*), realizando una estimación del esfuerzo necesario para cada tarea y asignándola a un miembro del equipo.

Durante el desarrollo de cada *sprint*, se realizan reuniones diarias (*Scrum daily meeting*) de corta duración donde se revisa el progreso realizado y los problemas surgidos desde la última reunión con el objetivo de realizar las adaptaciones necesarias para cumplir con la previsión de objetivos del final del *sprint*, basándose en el *Sprint Backlog* y actualizándose este a medida que se cumplen las tareas.

Al finalizar el *sprint* en curso, se realiza una reunión para revisar los objetivos/requisitos completados en esa iteración (*Sprint review*) y una reunión de retrospectiva (*Sprint Retrospective*) para analizar la manera de trabajar de los miembros del equipo durante el *sprint* finalizado, ver problemas surgidos durante el *sprint* y cómo solventarlos.

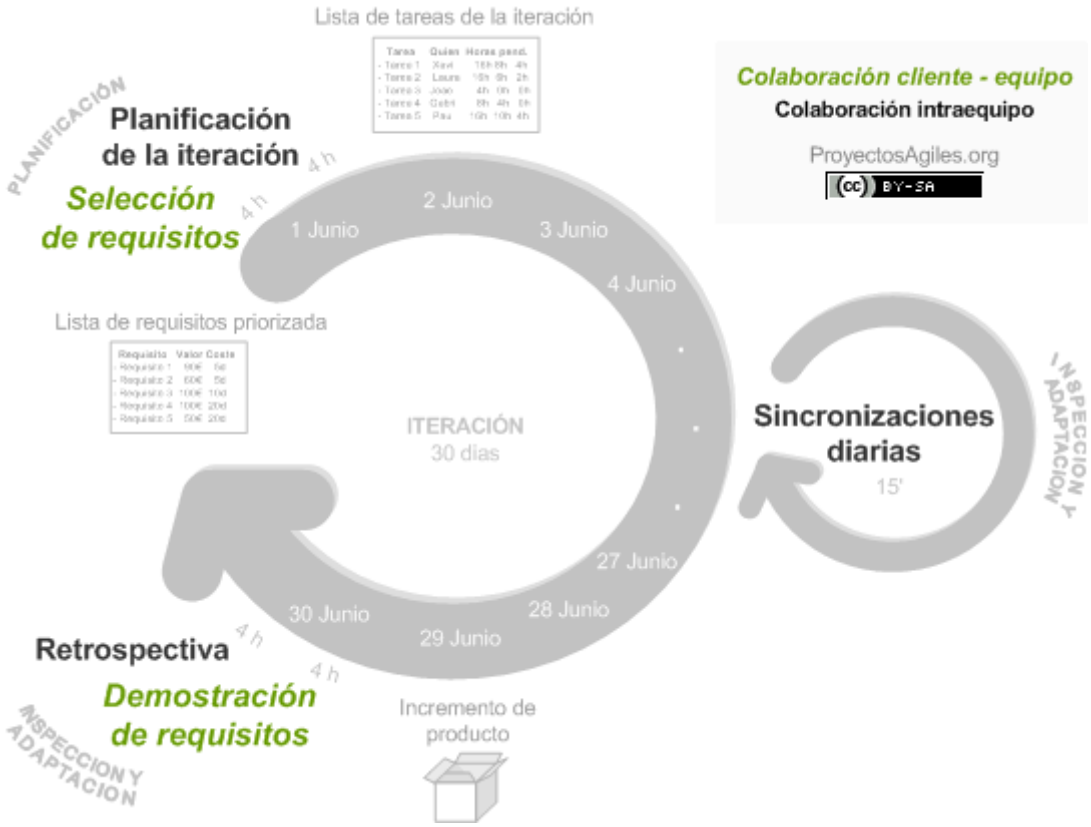

Figura 1. Esquema del proceso de un Sprint de Scrum (https://proyectosagiles.org/que-es-scrum/)

1.2.2.Scrum en el proyecto

Debido a las características particulares del proyecto y la disponibilidad de los miembros, se han realizado *sprints* semanales. Al inicio de cada semana realizábamos una única reunión que aunaba el *Sprint review*, el *Sprint Retrospective* y el *Sprint planning*. Las reuniones se realizaron a través de un video-chat al no poder hacerse las reuniones presenciales debido a la incompatibilidad de horarios entre los miembros del equipo. Su estructura era la siguiente:

1. Revisión de los objetivos realizados durante el sprint de la semana anterior y como se había abordado cada objetivo.

2. Exposición de problemas y dudas que se habían encontrado. En caso de no conseguir determinar el origen de un problema individualmente, se dedicaba un tiempo para su detección conjunta en el código fuente del proyecto.
3. *Sprint Planning*. Selección de objetivos/requisitos a implementar y división de tareas. Cada miembro hacía su propia lista de tareas para la culminación de cada objetivo asignado.

Así mismo, se realizaba una reunión diaria con la mínima duración posible (*Scrum daily meeting*), donde se realizaba un resumen de lo avanzado en el día y los objetivos personales que acometer para el día siguiente. En caso de detectarse un problema concreto en la realización del código fuente, se veía la mejor forma de abordarlo y corregirlo para permitir el avance del proyecto.

A lo largo de todo el proceso de desarrollo, se tenían reuniones con el tutor para definir algunas dudas y criterios para nuestra aplicación y para mostrar el avance en el proyecto.

En un inicio, las reuniones se centraron en la realización de los análisis de la competencia (sección 2.1 de la memoria) y de la identificación y elección de las herramientas de aprendizaje automático (sección 2.2 de la memoria). Estas reuniones fueron la base para establecer los diferentes requisitos de la aplicación, tanto funcionales como técnicos.

Una vez elegidas las técnicas y herramientas definitivas para el aprendizaje automático, aquí con TensorFlow, el trabajo se centró en su uso para la aplicación. Se abordó su utilización para el procesamiento de imágenes y su integración en un entorno basado en Android.

El primer paso fue comprender el proceso mediante el cual se generaba la red neuronal a través de un grafo computacional que permite realizar el proceso de clasificación de una imagen. Para ello se realizaron pruebas sobre colecciones de imágenes. Luego se procedió a una recopilación de imágenes de distintos edificios y monumentos de la UCM para realizar el grafo definitivo.

Con el grafo establecido, se procedió a realizar el desarrollo de una aplicación Android que integrara el sistema de clasificación. Debía permitir realizar la captura de una imagen, procesarla y mostrar el resultado en función de la clasificación.

Después, se procedió a añadir la geolocalización para la integración de un mapa. El objetivo era que el usuario tuviera una visión de conjunto de los elementos de interés conocidos en la aplicación.

Posteriormente se abordaron mejoras de la interfaz. Aquí se incluyeron mejoras visuales y de interacción, así como la integración de tecnología de síntesis de voz a partir del texto.

Para finalizar, se han realizado las dos evaluaciones mencionadas anteriormente. Una con usuarios reales y otra para la estimación de la precisión de clasificación obtenida finalmente.

Para la evaluación con usuarios reales, se ha pedido a conocidos que descarguen la aplicación desde la plataforma de Google Play, y la utilicen y la evalúen a través de una encuesta online. Únicamente se les ha indicado el propósito de la aplicación y los monumentos y edificios que podían probarse, pero no su funcionamiento. Los resultados de esta evaluación pueden verse en la sección 4.2.

Para la evaluación de la precisión en el reconocimiento de los monumentos, hemos realizado una serie de fotos desde diferentes ángulos a los edificios, estatuas y otros

monumentos que se pueden reconocer en Ciudad Universitaria. Después se ha procedido a evaluar el sistema según los valores devueltos por la API de TensorFlow.

1.3. Introducción al aprendizaje automático

El aprendizaje automático (*Machine Learning*) hace referencia a la detección automática de patrones significativos en los datos (Shalev-Shwartz & Ben-David, 2014). Es una rama del campo de la Inteligencia Artificial, que en palabras de Minsky (citado en Pajares & Santos, 2006, pp. 4-5) es "el estudio de cómo programar computadoras que posean la facultad de hacer aquello que la mente humana puede realizar". En otras palabras, conseguir que una máquina emule las funciones cognitivas de un ser humano.

En numerosas ocasiones, dada la complejidad de los patrones que es necesarios detectar, un programador humano no podría proporcionar una especificación detallada de todas las tareas que deberían ser ejecutadas (Shalev-Shwartz & Ben-David, 2014). Esta es la premisa por la cual nació el concepto del aprendizaje automático: buscar la detección automática de patrones significativos en los datos, de acuerdo con una cierta fórmula o algoritmo que permite clasificar ciertos datos o patrones de datos de entre un grupo mayor. Dicho algoritmo se ajusta mediante un entrenamiento o proceso de aprendizaje para optimizar el proceso de clasificación de los datos.

Los métodos del aprendizaje automático se pueden clasificar en dos grandes grupos considerando el tipo de estrategia y las ayudas que recibe el sistema, según Cambronero y Moreno (2006):

- Aprendizaje supervisado: permite la creación de un algoritmo predictivo a partir de unos datos iniciales de forma que permite distinguir la clase a la que pertenecen unos datos distintos a los iniciales. Según la documentación oficial de la herramienta de aprendizaje automático "Scikit-learn" disponible en el siguiente enlace: https://scikit-learn.org/stable/tutorial/basic/tutorial.html, el aprendizaje supervisado se divide, en función de su objetivo, en:
    - Clasificación: permite determinar la categoría de una muestra de datos de entre un conjunto de categorías predeterminadas. Por ejemplo, para la segmentación de un cliente, la categorización de imágenes y audio y el análisis de texto.
    - Regresión: permite obtener un resultado numérico que se determina en función de las entradas del modelo, en lugar de limitarse a un conjunto de etiquetas. Puede usarse para predecir resultados numéricos en función de datos. Por ejemplo, para predecir el importe económico que se obtendrá tras la realización de una campaña de marketing con ciertos criterios específicos.

- Aprendizaje no supervisado: permite el descubrimiento y presentación de estructuras en los datos mediante la agrupación de datos con similitudes, sin conocimiento previo del sistema.

En el caso de este proyecto, uno de los objetivos ha sido la detección de ciertos patrones en una imagen que permitan la clasificación de ese patrón en una categoría específica (el

monumento o estatua). De esta forma el sistema puede identificar dicha categoría y ofrecer información adicional sobre la misma.

Para concluir, en esta sección se han visto los objetivos específicos necesarios para poder dar por concluido con éxito la aplicación, así como la metodología de trabajo implementada y una pequeña introducción al concepto de aprendizaje automático y los tipos de metodologías aplicados en su implementación en función del tipo de estrategia y las ayudas que recibe el sistema durante el proceso de aprendizaje.

2. Estado del arte

En esta sección se procederá a un estudio del mercado y las herramientas disponibles en cuanto a aplicaciones relacionadas con la identificación de objetos y cómo podemos ampliar la funcionalidad de ellas con nuestro software. Se prestará especial atención a aquellas relacionadas con el turismo. La idea es siempre dirigir el trabajo a facilitar la información al usuario que realiza la vista guiada con la aplicación de forma rápida e intuitiva.

2.1. Análisis de la competencia

En esta sección se procederá al estudio de las características y funcionalidades de algunas aplicaciones existentes en el mercado que realicen tareas de reconocimiento de imágenes, sea en el ámbito del turismo o no.

- **Aplicación Google Lookout**: es una aplicación lanzada por Google en el 2018 diseñada para ayudar a personas ciegas o con problemas de visibilidad. La herramienta permite la detección e identificación de objetos mediante una cámara de ordenador o teléfono móvil, con la intención de describírselos al usuario en forma de voz. No solo reconoce el objeto, sino que también proporciona su posición. Por ejemplo "hay un perro a las 9 en punto". La aplicación tiene diversos usos: explorar el mundo real, facilitar la compra del usuario al leer códigos de barra y dinero, y leer textos. Es una app disponible en Google Play.

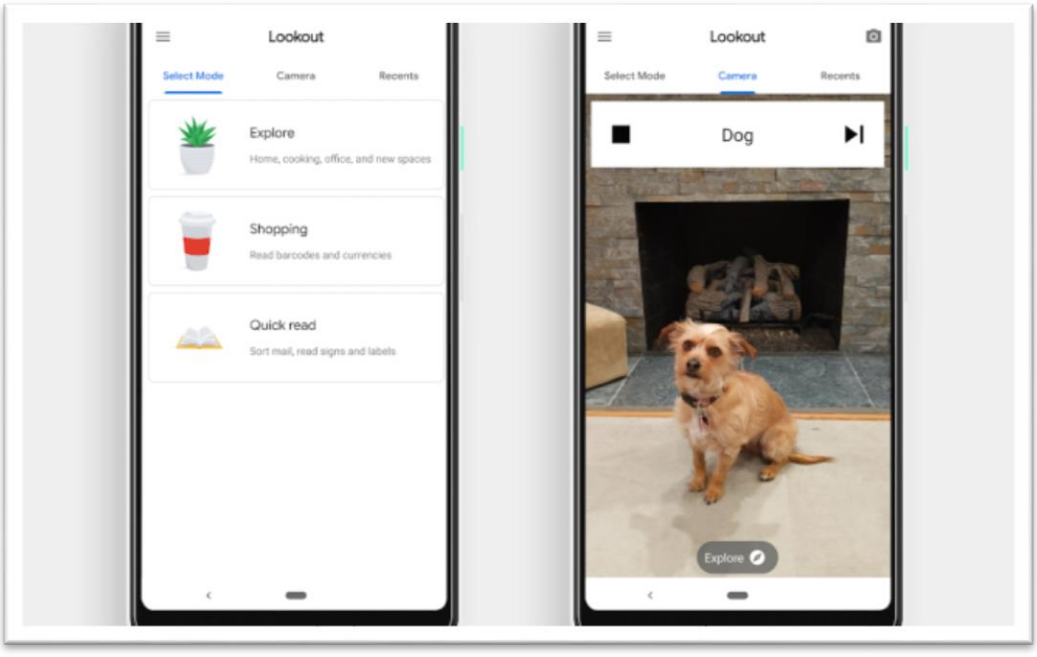

Figura 2. Captura de la app Lookout

Tras analizar la aplicación, se vio que proporcionar información sobre el contenido de la aplicación (ej. objeto identificado, descripción o texto) de forma visual y auditiva puede ser útil para adecuar el funcionamiento a diferentes situaciones de uso y usuarios.

- **Google Lens**: es una aplicación desarrollada por Google en el año 2018. Permite la captura e identificación de imágenes de diversos objetos, mostrando después resultados de búsqueda e información relevante encontrados en Google sobre el objeto identificado. Esta aplicación reconoce cierto tipo de objetos como monumentos, establecimientos, plantas y animales, comida y textos, entre otros. Actualmente Google la menciona como una versión de prueba.

  Tras utilizar la aplicación, vemos que solo muestra información de terceros, lo cual es incómodo para el usuario porque le obliga a navegar entre varias aplicaciones. Detectamos también un uso alto de la batería del teléfono móvil. Además, se ve por comentarios sobre la aplicación que no siempre identifica el objeto. Ello apunta a la necesidad de tener un amplio muestrario de imágenes de entrenamiento y la dificultad que entraña este proceso, incluso para una gran empresa como Google.

Figura 3. Captura de la app Google Lens

- **Aipoly Vision:** Se trata de una aplicación para el reconocimiento del entorno para usuarios con problemas de visión. Tiene una versión gratuita y otra de suscripción. Les permite reconocer objetos, colores (puede diferenciar hasta 1400 colores) y otras funcionalidades. En concreto reconoce elementos como mobiliario urbano, puertas o escaleras, y hasta comida o animales. Una vez lo reconoce lo nombra en voz alta. Su finalidad es aumentar la autonomía de las personas invidentes, procurando que puedan realizar actividades sin necesidad de ser acompañados por otra persona. Actualmente la aplicación se encuentra en desarrollo.

  Una vez hecho uso de la aplicación, se verificó la poca utilidad en nuestro contexto de tener una cámara constantemente reconociendo objetos. Además de la cantidad de energía usada, muestra una información tras otra en rápida sucesión con solo mover ligeramente el teléfono. Esto resulta molesto si lo que se desea es obtener información extensa sobre un objeto concreto.

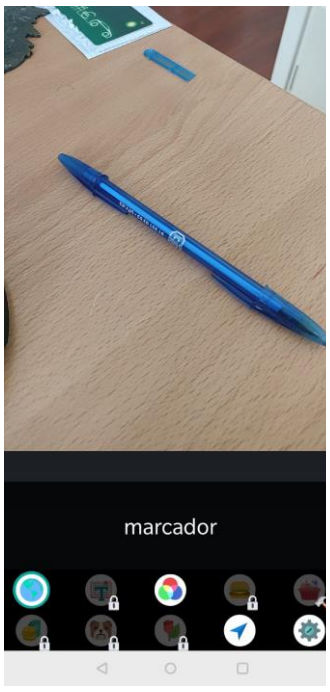

Figura 4. Captura de la app Aipoly Vision

- **San Lorenzo de El Escorial 360º:** es una aplicación de realidad virtual orientada hacia el ámbito del turismo. Permite el realizar un viaje virtual a ciertos puntos específicos de interés turístico de la localidad madrileña de San Lorenzo del Escorial, mediante la visualización de una recreación 3D de la localidad, pudiendo visualizarlos tanto en la actualidad como en el siglo XVIII. Esta aplicación ha recibido el Premio a la Mejor App Turística Nacional en la categoría de turismo temático otorgado por Segittur y FITUR en Madrid, en 2019.

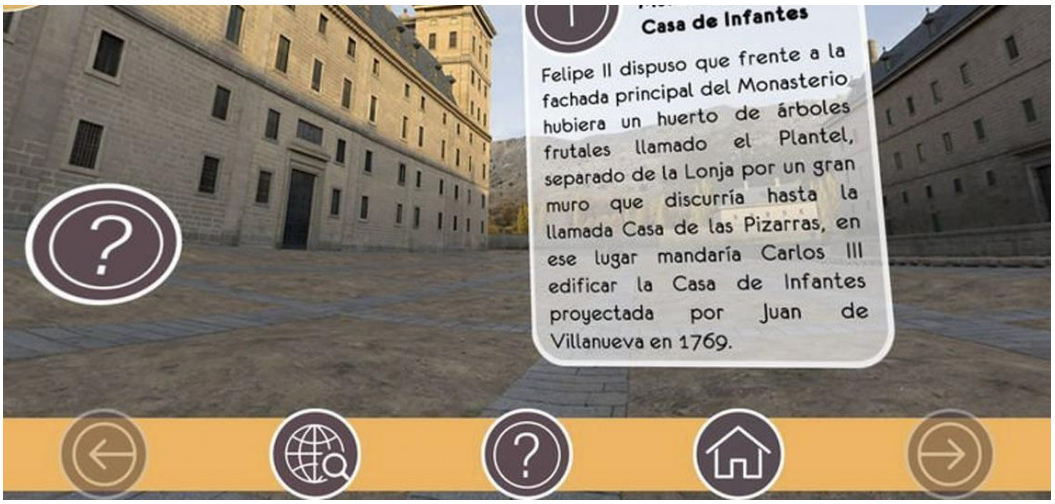

Figura 5. Captura de pantalla de San Lorenzo de El Escorial 360º

Tras probar la aplicación, fue evidente que añadir el entorno virtual a la aplicación ofrece una herramienta muy inmersiva al usuario. Sin embargo, presenta dos inconvenientes. En primer lugar, la necesidad de hacer una recreación 3D del lugar, con los costes que ello supone, y en segundo lugar la imposibilidad de moverse del lugar.

- **Quinta da Regaleira 4.0:** es una app diseñada por la empresa Orange Advertising para realizar rutas por la Quinta da Regaleira (Sintra, Portugal). Se trata de una aplicación de realidad aumentada. Permite el reconocimiento de ciertas imágenes, mostrando luego animaciones de distintos personajes mediante el uso de realidad aumentada para explicar la imagen vista. Reúne una serie de 40 animaciones, 30 vídeos y más de 100 audios en distintos idiomas (portugués, español e inglés).

Figura 6. Captura de la app Quinta da Regaleira 4.0

Tras analizar la app se descubrió el gran espacio que usa en el teléfono, pues una vez descargada es necesario ir descargando paquetes con videos y rutas. Además, la lentitud y la dificultad de reconocimiento de los objetos se destacaba bastante. Como punto positivo, introduce elementos de animación mediante realidad aumentada que hacen que la app sea muy entretenida, aunque en este caso, no por ello más útil.

- **Xátiva Turismo**: Es una aplicación que permite a los turistas que visitan la ciudad de Xátiva (Valencia) tener información acerca de los monumentos. Además, proporciona la localización de estos y la distancia que ha de recorrer el usuario hasta llegar al sitio de interés. Esta aplicación ha recibido el Premio a la Mejor App Turística Nacional en la categoría de turismo temático otorgado por Segittur y FITUR en Madrid, en 2018.

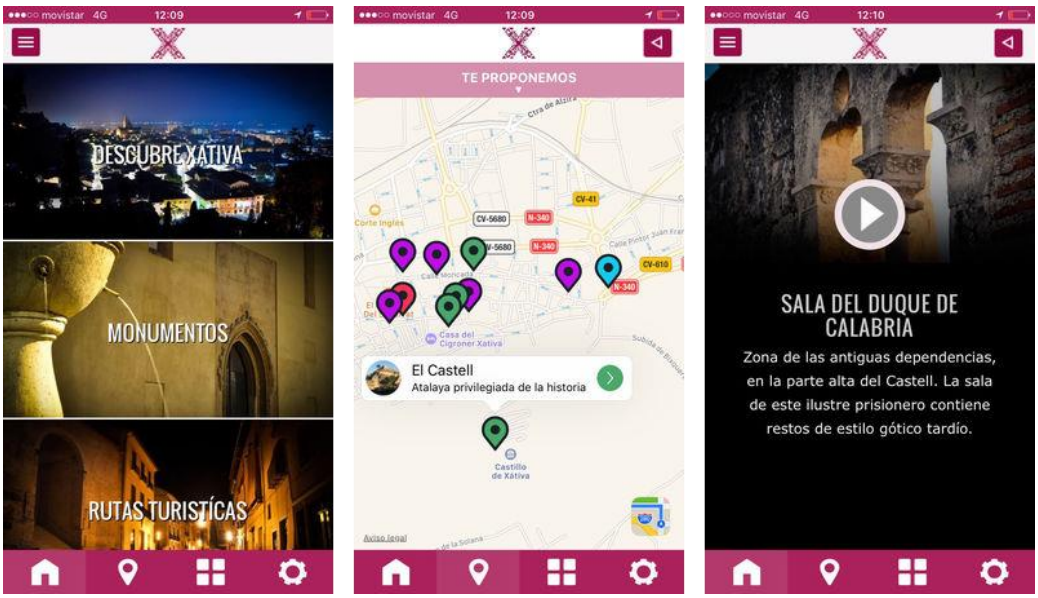

Figura 7. Capturas de pantalla de Xátiva Turismo

Después de utilizar la aplicación, vemos que la instalación es sencilla y la app es intuitiva y fácil de utilizar. Sin embargo, no dispone de reconocimiento de imágenes ni de entornos virtuales. Es una aplicación muy completa basada simplemente en mostrar información de la localidad con fotos, descripciones y ubicaciones. Se ve la utilidad de mostrar una lista de los sitios de interés, así como de mostrar dichos puntos en un mapa y ofrecer la opción de mostrar una información básica de cada lugar.

- **Phind:** es una aplicación de realidad aumentada que le permite al usuario recibir información de los objetos que tiene delante. Utiliza la cámara para captar la imagen y ofrece información al instante. Esta información puede ser un hecho relevante, una dirección, un teléfono de contacto, página web relacionada, comentarios de personas que hayan estado ahí con anterioridad e información de otros lugares que lo rodean. Para proporcionar esta información la aplicación utiliza el reconocimiento de imágenes y la localización, realizando un cruce de datos para obtener el resultado más probable.

Figura 8. Capturas de pantalla de Phind

Tras analizar la aplicación, vemos que tiene algunas dificultades para reconocer algunos lugares menos conocidos. Consume poca batería. Muestra la información al usuario de forma simple pero muy cómoda.

En base a las observaciones realizadas sobre las aplicaciones anteriores, se ha desarrollado la Tabla 1, que resume las características de cada una.

Tabla 1. Tabla comparativa de aplicaciones

	Lookout	**Google Lens**	**Aipoly Vision**	**San Lorenzo de El Escorial 360º**	**Quinta da Regaleira 4.0**	**Xátiva Turismo**	**Phind**
Realidad virtual o aumentada	Si	Si	Sí	Sí	Sí	No	Sí
Reconocimiento de Objetos	Sí	Sí	Sí	No	Sí	No	Sí
Texto a voz	Sí	No	Sí	No	Sí	Sí	No
Mapa	No, pero sí localización	No	No	Sí	No	Sí	Sí
Público Objetivo	Personas con dificultades de visión	General	Personas con dificultades de visión	Turismo	Turismo	Turismo	General
Ámbito turístico	No	No	No	Sí	Sí	Sí	Sí
Sistema operativo	Android	Android	iOS y Android	iOS y Android	iOS y Android	iOS y Android	iOS y Android
Información instantánea	Sí	Sí	Sí	Sí	Sí	Sí	Sí

2.2. Análisis de herramientas de *Machine Learning*

Existen multitud de plataformas, marcos y bibliotecas de aprendizaje automático en el mercado, tanto gratuitas como de pago, y de código abierto o privadas. Para el desarrollo de este proyecto, se analizaron algunos de los más populares, de código abierto, que ofrezcan estos servicios de forma gratuita, y con una comunidad de desarrolladores amplia. La Figura 9 recoge algunos de los marcos y bibliotecas de *machine learning* más usados en el momento de redactar esta memoria. Los más destacados se revisan a continuación en esta sección.

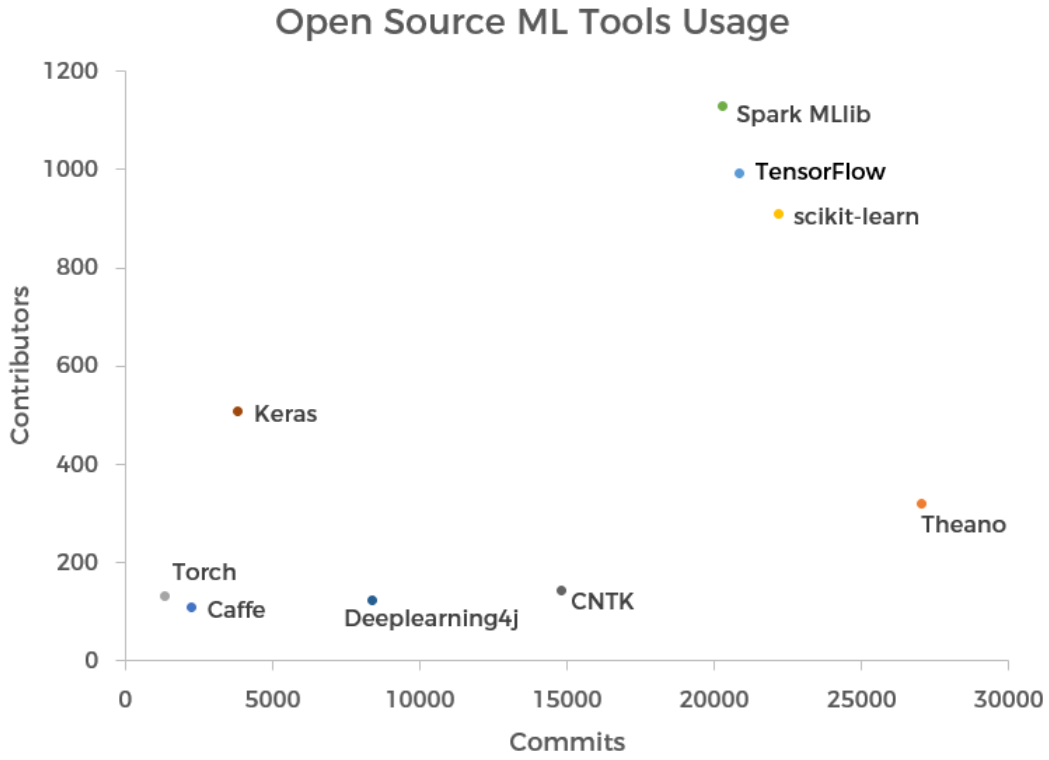

Figura 9. Comparación de commits y contribuidores para diferentes herramientas open source de machine learning en Github según la compañía AltexSoft en 2017 (https://www.altexsoft.com/blog/datascience/choosing-an-open-source-machine-learning-framework-tensorflow-theano-torch-scikit-learn-caffe/)

- **TensorFlow**: es un framework open-source de ML desarrollado por Google. Fácil de usar y desplegar en una amplia variedad de plataformas, es uno de los marcos más utilizados y populares de ML en la actualidad. Tiene un gran ecosistema de desarrolladores para su mantenimiento y ampliación. Posee una documentación muy exhaustiva.

  Es utilizado por varias compañías, entre ellas Dropbox, eBay, Intel, Twitter y Uber, Google Lens.

- **Scikit-learn:** es un framework open-source que fundamentalmente se centra en la minería y análisis de datos. Destaca entre otras características por la facilidad de uso y la documentación. Teniendo en cuenta su simplicidad y los numerosos ejemplos bien descritos, es una herramienta accesible para no expertos y que permite la aplicación rápida de algoritmos de aprendizaje automático a los datos.

  Scikit-learn ha sido adoptado por una gran cantidad de marcas exitosas como Spotify, Evernote, el gigante del comercio electrónico Birchbox y Booking.com.

- **Spark MLlib**: es la biblioteca de aprendizaje automático de Apache Spark. Su objetivo es hacer que el aprendizaje automático práctico sea escalable y fácil. En el momento de desarrollar este documento, se encuentra en proceso de migración a la versión de Spark 2.0.0, por lo que se decidió no ahondar en esta biblioteca.

- **Theano**: es una biblioteca open-source para computación científica basada en Python desarrollada por el Instituto de Montreal en 2007. Es una de las bibliotecas más antiguas y se considera un estándar dentro del campo de estos desarrollos. Originalmente diseñada para la definición, optimización y evaluación de expresiones matemáticas, especialmente aquellas con matrices multidimensionales.

  Es destacable la eficiencia, aunque los usuarios suelen quejarse de una interfaz poco intuitiva. Por ello suele usarse junto con otros marcos de alto nivel como Keras para facilitar esta tarea.

- **Keras**: es una biblioteca open-source de alto nivel diseñado para simplificar la creación de modelos de aprendizaje. Se puede implementar sobre otras tecnologías de inteligencia artificial como TensorFlow, Microsoft Cognitive Toolkit (CNTK) y Theano.

  Keras aumentó su popularidad debido a su facilidad de uso, modularidad y facilidad de extensibilidad, permitiendo la creación de prototipos de forma fácil y rápida.

La Tabla 2 resume algunos de los datos recogidos de los distintos marcos.

Tabla 2. Comparativa de características de herramientas de machine learning

	TensorFlow	Scikits-learn	Theano	Spark MLlib	Keras
Gratuito	Sí	Sí	Sí	-	Sí
Open source	Sí	Sí	No	-	Sí
Licencia	Apache License 2.0	BSD license	Copyright (c)	-	Apache License 2.0
Documentación	Sí	Sí	Sí	-	Sí
Proporciona APIs	Sí	Sí	Sí	-	Sí
Permite algoritmos de clasificación	Sí	Sí	Sí	-	Sí
Plataformas	Windows, Linux, Android, Iphone/IPad, MacOS	Windows, Linux, MacOS	Multi-plataforma	-	Linux, MacOS, Windows
Escrito en	C++, Python	Python, Cython, C y C++	Python	-	Python

2.3. Análisis de herramientas de BBDD

En esta sección se analizarán algunas de las bases de datos que se pueden utilizar para implementar la aplicación. Se dará una breve introducción a ellas y sus características más relevantes. Dividiremos estas bases de datos en función de la organización de la información:

- **Bases de datos relacionales**: se basan en un modelo Entidad-Relación. Usa una colección de tablas para representar elementos básicos (entidades) y las relaciones entre ellos (relaciones). Las estructuras de datos son rígidas y consistentes, cada campo con un tipo definido, por lo que es necesario realizar una planificación previa a la integración.

  Cumplen las denominadas características ACID, cuyas siglas vienen de las palabras en inglés: atomicidad, consistencia, aislamiento y durabilidad. Son propiedades que las bases de datos relacionales aportan a los sistemas y les permiten ser más robustos y menos vulnerables ante fallos.

  Algunos ejemplos son:

  - **MySQL**: es un sistema de gestión de bases de datos relacionales de código abierto desarrollada por Oracle Corporation (Kofler, 2001) bajo licencia GPL o Uso comercial. La instalación es sencilla y ofrece un entorno de fácil uso para la visualización y manejo de las tablas y bases de datos. En él se pueden realizar consultas a las bases de datos y gestionar los permisos de acceso de usuarios. La gestión de estos permisos requiere pasos de configuración poco intuitivos y complejos de gestionar.

  - **Microsoft SQL Server**: es una herramienta para la gestión de bases de datos desarrollada por Microsoft.

    Es software libre cuya integración con Android Studio es relativamente sencilla y que dispone de bases de datos con sincronización en tiempo real, es decir, que cualquier cambio realizado en los datos se sincroniza automáticamente y de manera inmediata a todos los usuarios, sin necesidad de que estos realicen una consulta, siempre que estén conectados con el servidor. En caso de no tener conexión, los datos se actualizarán en cuanto se produzca la siguiente conexión.

    El tema de seguridad en esta herramienta en cuanto al acceso de usuarios es difícilmente gestionable. Entre las dos opciones es la que da más problemas a la hora de asignar permisos de usuario o definir rangos.

- **Base de datos no relacionales**: se basan en un modelo orientado a documentos. Al contrario que una base de datos relacional en la que todos los registros deben tener los mismos atributos, incluso vacíos, a un documento no se le exige ajustarse a un esquema estándar, ni tener las mismas secciones, atributos o claves que otro, por lo que nunca quedan atributos vacíos. De este modo es posible añadir nueva información sin necesidad de establecer qué información queda excluida, por lo que son muy flexibles.

Normalmente cumplen solo algunas propiedades ACID para ganar eficiencia, realizando solo transacciones a nivel de documento y sin realizar un control sobre la integridad relacional.

Algunos ejemplos son:

- **Firebase Realtime Database**: es un sistema de gestión de bases de datos no relacionales desarrollado por Google y alojado en el servicio Firebase. Este servicio proporciona a los desarrolladores una API que permite almacenar y sincronizar datos en la nube en tiempo real, compartiendo una misma instancia de la base de datos y recibiendo actualizaciones automáticas al modificarse cualquier registro en la base de datos. Esta API está diseñada para permitir solo operaciones que se puedan ejecutar rápidamente.

  En caso de no tener conexión, los datos persisten en el disco, y se actualizan al conectarse de nuevo a la red para obtener la última actualización de los datos.

  La integración para el sistema operativo Android es la más sencilla encontrada.

  La seguridad del acceso de usuarios está gestionada por las cuentas de correo propias de Google. Esto hace que la seguridad de esta herramienta sea bastante sencilla de gestionar.

- **MongoDB**: es un sistema de gestión de bases de datos no relacionales orientada a documentos, cuyo código fuente está abierto bajo la licencia AGPL. Al estar orientado a documentos, no sigue un esquema definido dentro de una misma colección, con atributos o columnas presentes solo en algunos registros. Funciona en sistemas operativos Windows, Linux, OS X y Solaris.

  Sus características más destacadas son su velocidad y el sencillo sistema de consulta de los contenidos de la base de datos.

La Tabla 3 resume los datos recogidos de las distintas herramientas.

Tabla 3. Tabla comparativa de herramientas de BBDD

Propiedad	MySQL	Microsoft SQL Server	Firebase Realtime Database	MongoDB
Base de datos relacional	Sí	Sí	No	No
Sincronización en tiempo real	No	Sí	Sí	No
Uso gratuito	Sí	Sí	Sí	Si
Implementación sencilla	No	Sí	Sí	Sí
Relacional	Sí	Sí	No	No
Licencia	GNU General Public License	Comercial	Comercial con un plan gratuito.	AGPL
Transacciones	ACID	ACID	Sí	Transacciones ACID dentro del mismo documento
Triggers	Sí	Sí	Las *callbacks* se activan cuando los datos cambian en la BD	No

2.4. Conclusiones de los análisis

Aunque existen multitud de aplicaciones dedicadas al turismo, aún son pocas las que incorporan de forma eficaz el ML para el reconocimiento de edificios, monumentos y otros lugares de interés turístico. No obstante, esta tecnología está en auge y el número de aplicaciones relacionadas va en aumento.

Tras analizar las aplicaciones disponibles en el mercado, se han establecido las características mínimas obligatorias que se deben tener, las deseables y las que se implementarán en un futuro desarrollo de la aplicación. Estas son:

- Obligatorias:
  - Captura e identificación de objetos a partir de una imagen. Se debe capturar en un momento especifico mediante una cámara para evitar el uso excesivo de la batería.
  - Interfaz sencilla e intuitiva.
  - Mostrar la lista completa de los lugares de interés.
  - Mapa para mostrar la ubicación de los puntos de interés.

- Deseables:
    - Sistema de conversión de texto a voz para describir los detalles de puntos de interés de forma auditiva. Ello dará una mayor comodidad al usuario y hará la aplicación más inclusiva para usuarios con diversidad funcional.

- Mejoras futuras:
    - La realidad virtual o la realidad aumentada se incorporará en futuras implementaciones, pero no para la versión original, debido al alto coste en tiempo que supondría este tipo de desarrollo.
    - Optimización de los algoritmos de clasificación.
    - Ampliación de la información histórica de los edificios y monumentos con fuentes oficiales.

Por otro lado, tras realizar el análisis en la sección 2.2 sobre las distintas herramientas de ML, se ha decidido utilizar TensorFlow. Las razones incluyen:

- Permite la clasificación a partir de un conjunto de datos en forma de imágenes de forma sencilla.
- Facilita APIs para la integración en una plataforma Android.
- Permite crear aplicaciones móviles ligeras. Uno de los proyectos de Tensorflow, MobileNet, desarrolla conjuntos de modelos de visión por computador diseñados para abordar las ventajas y desventajas de velocidad / precisión considerados en dispositivos móviles o aplicaciones integradas.
- La amplia documentación, herramientas y ejemplos que se ofrecen. Esto permitirá un rápido avance en la asimilación de esta tecnología.
- El gran número de desarrolladores que lo respaldan.

Será por tanto el *framework* para realizar la clasificación de edificios y monumentos de la UCM.

Finalmente, en base al análisis realizado en la sección 2.3 se llegó a la elección de Firebase Realtime Database como base de datos. Se trata de la solución que mejor se adapta a las necesidades de desarrollo de la aplicación debido a los siguientes motivos:

- **Flexibilidad en la estructura de la base de datos**. Al no conocer el alcance del proyecto debido al desconocimiento sobre la dificultad de integración de la tecnología de reconocimiento de objetos, necesitaremos una base de datos que se adapte rápidamente a los cambios que puedan surgir en el proyecto. Al ser una base de datos no relacional, permite realizar esos cambios con facilidad.

- **El sistema de sincronización de datos que ofrece Firebase**. Éste permite que los datos de la aplicación estén actualizados en todo momento, viendo los cambios casi al instante al modificarse en la base de datos. Si un usuario no cuenta con conexión a la red, sigue pudiendo ver la información relacionada con un edificio o monumento. Esta información se actualizará cuando vuelva a conectarse a la red.

- **Implementación sencilla**. La implementación de la API de Firebase en una aplicación Android es muy sencilla. Así mismo, podemos usar cualquier URL de Firebase Realtime Database como un extremo de REST o modificar directamente los datos desde la interfaz que proporciona el propio Firebase.

- **Gran cantidad de documentación accesible al desarrollador**.

- **Seguridad:** La seguridad de acceso y modificación a los datos está gestionada por Google a través de las cuentas personales propias de Google.

3. Sistemas relacionados

En este punto se describirán los sistemas relacionados con la aplicación realizada. Los sistemas relacionados son tecnologías ya desarrolladas cuyo uso se considera adecuado o necesario para el desarrollo de este proyecto.

Tras haberse realizado un estudio de campo sobre diversas tecnologías (ver sección 2), las finalmente escogidas fueron TensorFlow y Firebase, ambas de Google. Así mismo, se hará una introducción a Google Maps. A continuación, se procederá a introducir estas tecnologías y explicar brevemente su funcionamiento.

3.1. TensorFlow

3.1.1. Introducción

TensorFlow es una librería de Google de código abierto destinada al cálculo numérico. Para realizar estos cálculos TensorFlow hace uso de grafos computacionales cuya estructura es la siguiente:

- Nodos del grafo: Los nodos de estos grafos son operaciones matemáticas que reciben parámetros de entrada y retornan una única salida.

- Aristas: Las aristas o conexiones entre nodos son llamadas *tensores* y representan conjuntos multidimensionales de datos.

El nombre de TensorFlow viene de los *tensores* sobre los que se realizan las operaciones de los nodos. Uno de los principales usos de esta librería es el de entrenar redes neuronales con el fin de encontrar patrones en datos.

Como se mencionó al inicio, esta librería pertenece a Google pero no fue hasta el 2015 que fue considerada código abierto. Puede ser integrada en múltiples plataformas como iOS, Android, MacOS, Linux o Windows.

3.1.2. Funcionamiento TensorFlow

TensorFlow permite implementar redes neuronales para el reconocimiento de imágenes. En primera instancia, una neurona es una función matemática que calcula la suma ponderada de ciertas entradas que se le proporcionan. Cada entrada tiene un peso o importancia que contribuyen al valor resultante en su única salida. Además de estas entradas, existen funciones de activación que evitan que el resultado de sumar varias neuronas dé una regresión lineal.

La salida de una neurona se puede usar como entrada de otra. De esta forma se pueden realizar varias capas de neuronas conectadas entre sí construyendo una red. Las entradas externas a la red alimentan las primeras neuronas cuyos resultados se propagan a las siguientes. Este procedimiento continúa hasta la salida final de la red.

Una vez obtenido el resultado final puede comenzar el entrenamiento. Si es un resultado desfavorable, se emplea el método de *backpropagation*, que consiste en revisar la capa

anterior de neuronas, asignarles un peso de error a cada una en base a cuál ha tenido más influencia a la hora de producir el error y, mediante este peso, ajustar los valores de peso a la entrada de las neuronas. Este proceso es recursivo.

3.2. Firebase

Firebase es una plataforma de servicios en la nube que facilita la implementación de varios servicios de Google mediante un mismo SDK. Entre los servicios destacados de Firebase están Firebase Analytics, Firebase Cloud Messaging y Firebase Storage. Este último es el servicio de base de datos en Real Time.

Este tipo de base de datos permite un tratamiento en "tiempo real" de la información proporcionada en la aplicación. En la terminología de Firebase esto quiere decir que los usuarios usan las últimas modificaciones disponibles en la base de datos sin tener, en el caso de una aplicación Android, que reiniciar, actualizar ni cambiar de Activity. Se optó por emplear este tipo de sistema por los siguientes motivos:

- Fácil y rápida escalabilidad.
- Base de datos no SQL.
- Tiempo de respuesta óptimo.
- Implementación más sencilla frente a una base de datos relacional.

3.3. Google Maps

Google Maps es un servicio web desarrollado por Google que facilita información geográfica de todo el mundo. Dispone de varias funcionalidades entre las cuales se encuentran la visualización aérea del terreno, vistas en 360º de muchas ciudades, y mapas de carreteras convencionales. Esta última funcionalidad permite trazar rutas de camino mínimo de un punto a otro de una ciudad.

Google Maps ofrece a los desarrolladores múltiples servicios a través de su API. Entre las más usadas se cuenta marcar ubicaciones específicas en sus mapas y planificar rutas para conductores o transeúntes que desean viajar desde su ubicación actual a otro punto específico del mapa. Estas son las funcionalidades usadas en este trabajo.

4. Implementación

En esta sección se comentarán y explicarán las diferentes tecnologías que componen la arquitectura del sistema. Cada punto se convierte en uno de los componentes que conforman la aplicación y en ellos se indica tanto su funcionamiento como su papel en el resultado final.

El objetivo es generar y usar una red neuronal de reconocimiento de imágenes, que se representa mediante un grafo computacional. En la primera parte se explica el funcionamiento general de la aplicación con un diagrama de actividad explicativo, seguido de la arquitectura que empleamos de MVC y los diagramas de clase de la aplicación.

Finalmente se explican tecnologías que fueron empleadas en el desarrollo de la aplicación como TensorFlow, Firebase y Google Maps.

Mencionar también el uso de GitHub como herramienta de control de versiones del código fuente y Google Drive para compartir la documentación e imágenes utilizadas durante la implementación de este proyecto.

4.1. Funcionamiento de la aplicación

En esta sección se mostrarán esquematizados los diagramas de clases de las actividades de reconocimiento de imagen y gestión del mapa. También se explica el funcionamiento general de la aplicación.

4.1.1. Funcionamiento general

La aplicación consiste en una ayuda a los turistas que recorran Ciudad Universitaria. Una vez inicien la aplicación, podrán seleccionar la opción de *reconocer monumento* la cual abre la cámara del dispositivo móvil y permite tomar una fotografía a un monumento. Una vez tomada, el programa intenta reconocer el monumento y devuelve un valor entre cero y uno que representa la precisión en porcentaje que tan seguro está de haber reconocido la imagen. Si el porcentaje es inferior a ochenta y cinco por ciento, se deduce que no ha reconocido el monumento y se muestra un mensaje de objeto no reconocido.

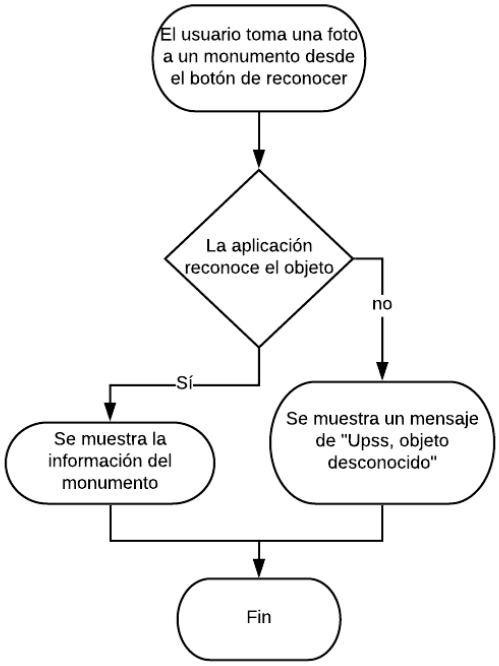

Figura 10. Diagrama del reconocimiento de imagen

4.1.2.Diagramas de clases

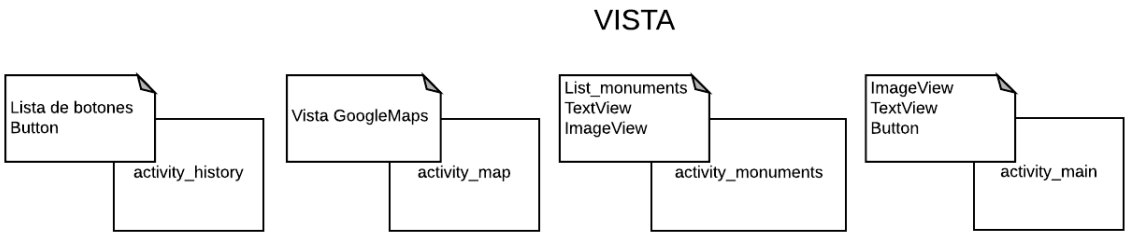

Figura 11. Diagrama de clase de la capa de vistas

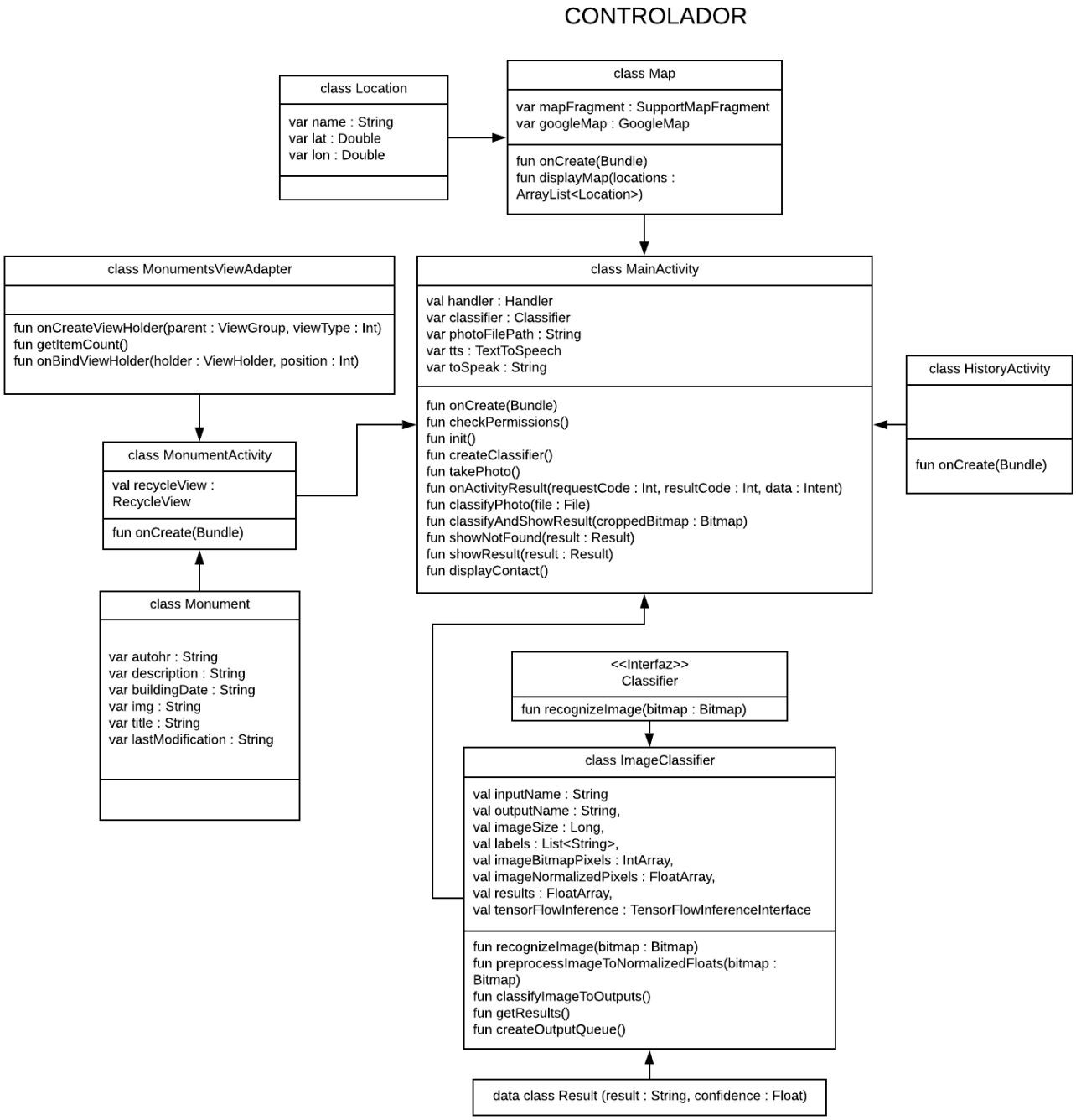

Figura 12 Diagrama de clase de la capa Controlador

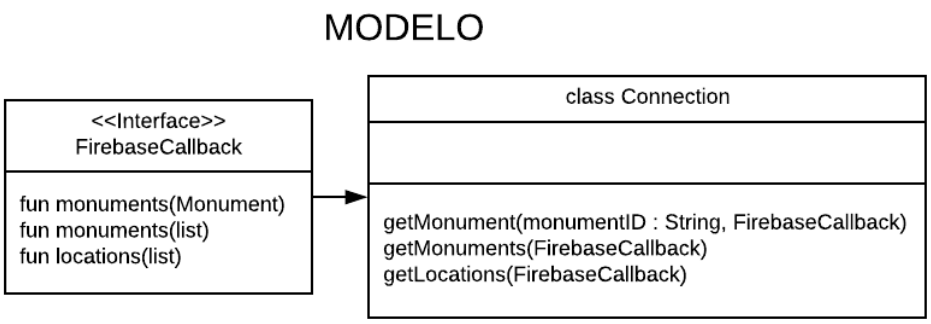

Figura 13 Diagrama de clase de la capa Modelo

4.2. Arquitectura

Como explican Sauter, Vögler, Specht y Flor (2005) en su libro, el Modelo-Vista-Controlador es un patrón de arquitectura empleado para el desarrollo de aplicaciones software que se encarga de separar la lógica de negocio de la interfaz de usuario. Este patrón ayuda a tener varias vistas (visualización de datos), varios controladores (orquestación) y relacionarlos con un único modelo (gestión de datos). Esta es la arquitectura que se siguió en todo el proceso de desarrollo de este software.

Para entrar en profundidad, se definirán cada una de las capas que conforman este patrón de arquitectura representadas en la Figura 14. (González & Romero, 2012).

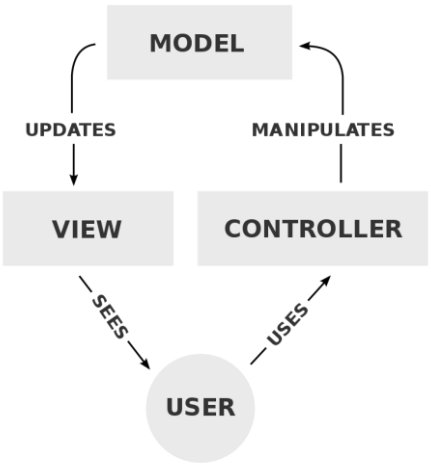

Figura 14. Ejemplo de estructura Modelo-Vista-Controlador

4.2.1.Modelo

Representación de la información que emplea el software. Esta información generalmente se presenta en forma de base de datos la cual tendrá una estructura configurada por el desarrollador de la aplicación.

El modelo se encarga de gestionar todos los accesos a la información, de realizar las consultas pertinentes y de implementar las manipulaciones de datos y los privilegios de acceso que se hayan descrito en las especificaciones del software. Este último punto se conoce como lógica de negocio.

El usuario realiza peticiones al controlador, las cuales son gestionadas por éste y transmitidas al modelo. Una vez la petición llega al modelo, esta capa se encargará de enviarle a las vistas aquella parte de la información solicitada.

4.2.2.Vista

Representación visual de los datos de la aplicación, interfaces gráficas y, en resumen, parte del software con la que interactúa el usuario.

En esta capa se recoge la información recibida de la capa de modelo y es presentada de forma legible y amigable para el usuario. Se compone generalmente de texto, imágenes, campos donde introducir información, botones, etc.

4.2.3.Controlador

Esta es la capa que orquesta la aplicación y en particular las interacciones entre vistas y modelos. En ella se encuentra gran parte de la lógica de funcionamiento del software y se encarga de responder a las peticiones que realiza el usuario. Estas peticiones se gestionan y después se llama a la capa de modelo cuando se requiera consultar, crear, modificar o eliminar cierta información.

Otra función es enviar comandos a su vista asociada para realizar cambios en la forma en que se presenta la información recibida por el modelo, como por ejemplo hacer zoom o un *scroll*.

4.2.4.Estructura de la aplicación

A continuación, se explica la forma en la que se implementó la estructura Modelo-Vista-Controlador dentro de la aplicación y sus componentes de forma detallada.

Modelo / Model

- Connection

Clase encargada de realizar una conexión a la base de datos de Firebase y realizar las consultas necesarias. Esta clase consta de métodos o queries que se realizarán de forma independiente, logrando un mayor desacoplamiento del código y flexibilidad en cuanto a escalabilidad.

- `getMonument:` Recibe un ID (monumentID) de monumento y un callback. Consulta el monumento en la base de datos y lo devuelve por el callback.

- `getMonuments:` Consulta todos los monumentos de la base de datos de Firebase y devuelve una lista de tipo `Monument` por el callback.

- `getLocations:` Consulta todas las localizaciones de la base de datos de Firebase y devuelve una lista de tipo `Locations` por el callback.

- FirebaseCallback

Interfaz encargada de proporcionar los callbacks necesarios para las consultas o queries a la base de datos.

- `monument:` Callback empleado en la clase `Connection` que devuelve un objeto Monument.

- `monuments:` Callback empleado en la clase `Connection` que devuelve lista de objetos Monuments.

- `locations:` Callback empleado en la clase `Connection` que devuelve lista de objetos `Location`.

Vista / View

- MainActivity

Clase `Main` encargada del cuerpo principal de la aplicación. Presentará una pantalla con un menú de opciones y el botón para reconocer los monumentos de una manera más accesible.

- `checkPermissions:` Método que pide y acepta los permisos necesarios para acceder a la cámara del dispositivo y al almacenamiento para una posterior recuperación de las imágenes.

- `createClassifier:` Crea el objeto `ImageClassifier` al que le pasa como atributo el grafo entrenado y el archivo con los nombres de los objetos a reconocer.

- `takePhoto:` Acción que permite al usuario tomar una foto, saltar a la pantalla de aceptar o reintentar y en caso de aceptar, empieza el proceso de reconocimiento.

- `classifyPhoto:` Recupera la imagen tomada por el usuario transformándola en bitmap para finalmente pasársela al método de clasificar.

- `classifyAndShowResult:` Genera un hilo en *background* que usa el objeto `classiffier` para reconocer la imagen anteriormente tomada por el usuario.

- `showResult:` Muestra toda la información del monumento al que se le tomó la foto junto con una descripción y la posibilidad de escucharla en audio.

- MonumentsActivity

Vista que muestra una lista de los monumentos disponibles para visitar en forma de álbum de fotos, cada uno de ellos con el nombre correspondiente. Se empleó un `viewAdapter` ya que el contenido visual se almacena en la base de datos. En caso de añadir nuevos monumentos en la base de datos, esta lista se actualizará en tiempo real.

- MonumentsViewAdapter

`ViewAdapter` que genera la estructura para la visualización de la lista de monumentos. La estructura se basa en un texto con el nombre del monumento o edificio con una posterior representación gráfica del mismo. Esta vista, con ayuda del `RecycleView`, mostrará todos los monumentos que se encuentren en la base de datos.

- Map

Vista que carga de la base de datos todas las localizaciones de los monumentos disponibles para el tour y los muestra en un mapa. Para la realización de esta vista se contó con la API de Google Maps, la cual ofrece servicios como personalizar los puntos de interés donde se encuentren los monumentos, mostrar los nombres de los mismos y trazar la ruta más corta hasta ellos.

- `displayMap:` Una vez cargadas las localizaciones, llama al objeto *googleMap* el cual las añade al mapa y centra la vista en las coordenadas donde se encuentra la Facultad de Informática de la UCM en este caso.

- HistoryActivity

En esta vista se muestran enlaces con información a cada uno de los momentos históricos que hay en la Ciudad Universitaria, incluyendo su origen, sus etapas importantes y su crecimiento.

Controlador / Controller

- Location

Objeto que refleja los atributos de una localización. Los atributos son; nombre de la ubicación, latitud y longitud. (name, lat, lon).

- Monument

Objeto que refleja los atributos de un monumento. Los atributos son; autor, descripción, fecha de construcción, url de la imagen, título del monumento y fecha de última modificación. (autor, description, buildingDate, img, title, lastModification).

- ImageClassifier

Objeto que procesa la clasificación de una imagen mediante un grafo entrenado y un archivo de etiquetas llamado *labels.txt*. Los atributos de la constructora son el nombre de

entrada o *input*, el tamaño de la imagen, las etiquetas del archivo *labels.txt* o nombre de los monumentos a reconocer y una inferencia del TensorFlow.

- `preprocessImageToNormalizedFloats:` Este método procesa la imagen en enteros de 0 a 255 y lo normaliza en base float.

- `classifyImageToOutputs:` Método que alimenta el grafo de TensorFlow, lo corre y guarda un resultado en el outputName.

4.3. TensorFlow

En esta sección se explica detalladamente como emplear TensorFlow para entrenar una red neuronal capaz de reconocer imágenes proporcionadas por el usuario. Es imprescindible contar con estas imágenes, las cuales se guardarán en un repositorio central y no en los clientes móviles. Esta estructura se explicará más adelante.

4.3.1.Entrenamiento y generación del grafo computacional

Puesto que todo el entrenamiento se realiza mediante scripts de Python, este punto se desarrollará completamente con el uso de la consola de comandos.

En primera instancia, y desde la consola de comandos, se accederá a la carpeta donde se instaló Python. La ruta será algo semejante a lo siguiente: *C:\Users\user\AppData\Local\Programs*. En esta ruta se creará un entorno virtual, es decir, un entorno de Python que incluye un árbol de directorios, así como el intérprete y las bibliotecas propias de Python y que permite aislar los scripts contenidos en los directorios de otros entornos virtuales. Para la creación, podemos consultar la sección 10.1.

Dentro de este entorno se deberá crear una carpeta por cada tipo de imagen (categoría) que se quiera reconocer. En cada carpeta se guardarán todas las imágenes del mismo tipo que guardan relación. Un punto importante a tener en cuenta es el hecho de que no se podrá entrenar un único tipo de imagen. Es imperativo tener como mínimo dos ya que TensorFlow necesita un punto de comparación binario. La estructura deberá ser semejante a la mostrada en la Figura 15**¡Error! No se encuentra el origen de la referencia.**.

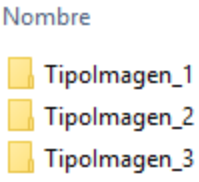

Figura 15. Estructura de carpetas para las imágenes de entrenamiento

El siguiente paso será activar el entorno virtual con el script `activate` como se mostró en la sección anterior. Una vez se confirme su activación, estará todo listo para el proceso de entrenamiento.

Para el proceso de entrenamiento se hará uso del script `retrain.py` que se encuentra en la carpeta *Scripts* dentro del entorno virtual. Para ello se empleará el siguiente comando:

```
python .\Scripts\retrain.py --image_dir .\dir_de_carpeta_con_
_imagenes --architecture mobilenet_1.0_224 --
how_many_training_steps=500
```

Este comando ejecutará el script *retrain.py*, busca en la carpeta donde se han guardado las imágenes, elije la arquitectura `mobilnet`, puesto que el destino será una aplicación móvil, y finalmente asigna 500 pasos de entrenamiento para la red neuronal.

Tras ejecutar el anterior comando, se generará una carpeta *tmp* en el directorio *C:* donde se guardará el grafo entrenado bajo el nombre de *output_graph.pb* y un archivo con todos los nombres de las carpetas de la estructura de imágenes que se creó al inicio llamado *output_labels.txt*. Este último archivo servirá de id para reconocer las imágenes que se han entrenado. No modificar ninguno de estos archivos.

Para comprobar que el grafo generado es correcto se intentará ejecutar un reconocimiento de una de las imágenes empleadas para su entrenamiento. El resultado deberá salir con un porcentaje alto, próximo a 0.9% según se ha obtenido en las evaluaciones de prueba (ver Sección 6.2). El comando para reconocer una imagen es el siguiente:

```
python .\Scripts\label_image.py --graph=C:\tmp\output_graph.pb -
-labels=C:\tmp\output_labels.txt --output_layer=final_result --
image=.\ dir_de_carpeta_con_
_imagenes\TipoImagen_1\nombre_imagen.jpg --input_layer=input.
```

4.3.2.Proceso de reconocimiento dentro de la aplicación

En primera instancia, al iniciar la aplicación y aceptar los permisos de acceso al contenido audiovisual del terminal, se crea un objeto `ImageClassifier` el cual obtiene como parámetro de su constructora el grafo generado previamente en el entrenamiento con TensorFlow y el archivo de etiquetas llamado *labels.txt*. Con estos dos archivos se podrá reconocer imágenes asociándolas con las etiquetas encontradas en *labels.txt*.

Tras esto, la aplicación ofrece al usuario una pantalla inicial que le sugiere reconocer un monumento. Esto se realiza mediante una foto que el usuario debe tomar del monumento a identificar. Tras tomar la foto se le presenta al usuario la posibilidad de rechazar la imagen o aceptarla para proceder con el reconocimiento.

Una vez almacenada la imagen es pasada como parámetro al método `classifyPhoto` de la clase `Main`, el cual se encargará de transformar el archivo a un Bitmap. Un Bitmap o mapa de bits es una matriz de pixeles, los cuales tienen cada uno asignados un código de color. No obstante, el objetivo no es este mapa de bits, sino una versión recortada o *croppedBitmap*. Esta última versión del archivo tratado se pasa como parámetro al método `classifyAndShowResult`.

El método `classifyAndShowResult` ejecuta un hilo en *background* que llama al objeto `ImageClassifier` y emplea su método `recognizeImage`. A este método del `ImageClassifier` se le pasa como parámetro la variable *croppedBitmap* generada

anteriormente y devolverá un objeto `result`. Antes de entrar en detalle en el objeto `result` y como se emplea, se explicará cómo funciona el método `recognizeImage`.

El método `recognizeImage` llamará al método `preprocessImageToNormalizedFloats` cuya función será procesar los datos enteros de la imagen que son de 0 a 255 con el fin de normalizarlos en base *float*. Finalmente se llamará a `classifyImageToOutputs`. Este último método hará uso de una variable *tensorFlowInference* que se obtendrá de la librería de TensorFlow llamada *org.tensorflow.contrib.android.TensorFlowInferenceInterface*.

La variable *tensorFlowInference* llamará a tres métodos, *feed*, *run* y *fetch*. Estos métodos son los encargados de alimentar el grafo con la imagen tratada, de ejecutar el algoritmo y finalmente de retornar un valor de salida respectivamente. Este valor de salida se devuelve hasta llegar al anteriormente mencionado objeto `result`.

El objeto `result` no es más que un *data class* con dos atributos, un *val result : String* y un *val confidence : Float*. Estos valores serán respectivamente una de las etiquetas que se encuentran en el archivo *labels.txt* y un valor entre 0 y 1 que representa el umbral de aceptación de esa etiqueta.

Finalmente, con una etiqueta, la cual tendrá el mismo valor que uno de los hijos del elemento *monuments* de la base de datos, y un porcentaje de acierto, la aplicación podrá mostrar la información pertinente del monumento en cuestión.

4.4. Firebase

Esta tecnología es parte crucial de la capa de Modelo de la aplicación. En ella se realizará el tratamiento de la información, así como las consultas requeridas. Para su instalación se pueden seguir los pasos del siguiente enlace:
https://firebase.google.com/docs/android/setup?hl=es-419

4.4.1.Creación de la Base de datos

Una vez se ha agregado Firebase al IDE de Android Studio, se accederá a la consola de Firebase que se encuentra en el siguiente enlace: *https://console.firebase.google.com/?pli=1* . En esta pantalla se seguirán los pasos para la creación de un nuevo proyecto.

Figura 16. Consola de Firebase

Tras haber creado el proyecto se podrá acceder a la consola de Firebase donde se procederá a la creación de la base de datos. En esta aplicación se empleó Realtime Database para la gestión de la información de CIU Tour.

En el menú lateral se selecciona “Database” para acceder al panel de control de las bases de datos de Firebase. Se selecciona el tipo de base de datos (como anteriormente se indicó, Realtime Database) y finalmente en el botón de la cruz se puede crear la base de datos en un formato JSON como se observa en la imagen. También se pueden importar JSON propios.

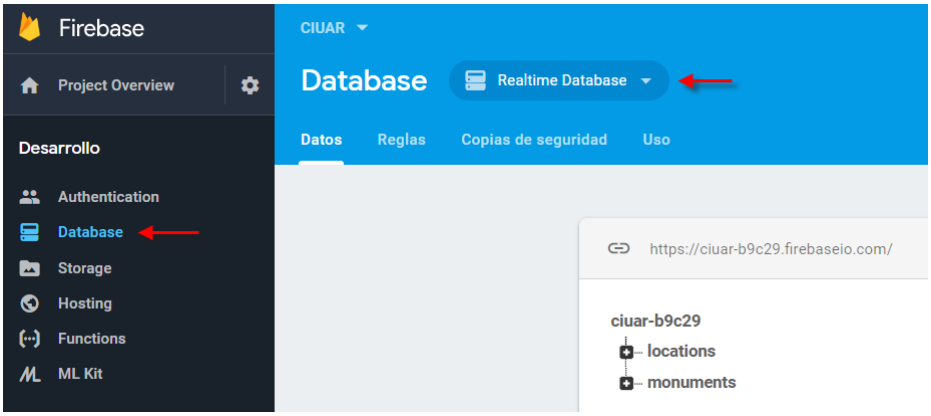

Figura 17. Panel de control de Firebase

4.4.2.Estructura

El nombre de la base de datos empleada es "ciuar-b9c29". La estructura de esta base de datos se generó mediante un JSON compuesto por dos elementos principales; *locations* y *monuments*. A continuación, se explicará en detalle cada uno y sus correspondientes hijos.

Locations

En este elemento se guardarán las localizaciones de los monumentos accesibles en el tour por Ciudad Universitaria. Cada localización está compuesta por su id, en este caso el propio nombre del monumento, y tres atributos esenciales que son su latitud, longitud y el nombre de la ubicación a la que hace referencia, como se puede apreciar en la Figura 18.

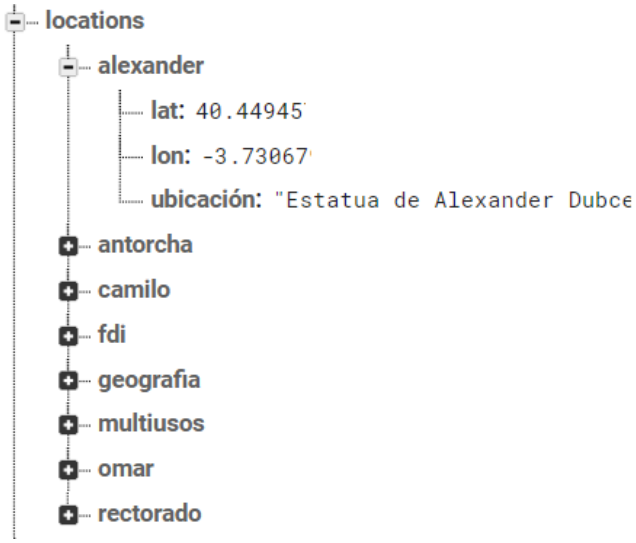

Figura 18. Tabla Locations

Gracias a la API de Google Maps, se podrá cargar esta información en la aplicación y mostrar en un mapa en tiempo real todas las localizaciones de los monumentos que se podrán visitar. Una vez desplegados los monumentos en el mapa se podrá pulsar en uno de ellos para observar el nombre de este y acceder a la posibilidad de mostrar la ruta más corta hasta él.

Monuments

En este apartado de la base de datos se guardará una lista de monumentos, cada uno con la información que lo representa. Esta información será de fácil y rápido acceso para los usuarios gracias a la conectividad en tiempo real de Firebase.

La estructura de esta tabla se puede observar en la Figura 19. Por cada monumento se podrá obtener su id, en este caso el propio nombre del monumento, seguido de los siguientes atributos:

- **autor**: Nombre del autor responsable de la obra o construcción del monumento.
- **descripción**: Descripción que se mostrará al usuario tras reconocer la foto que realizó. Esta información podrá tanto leerse como escucharse.
- **fechaConstruccion**: Fecha de construcción del monumento o edificio.
- **imagen**: Ruta URL donde se almacena una imagen del monumento. Esta imagen se guarda en el propio Firebase en el apartado Storage como se observa en la Figura 20.
- **titulo**: Título o nombre del monumento en cuestión.
- **ultimaModificacion**: Fecha de su última modificación o restauración.

Figura 19. Tabla Monuments

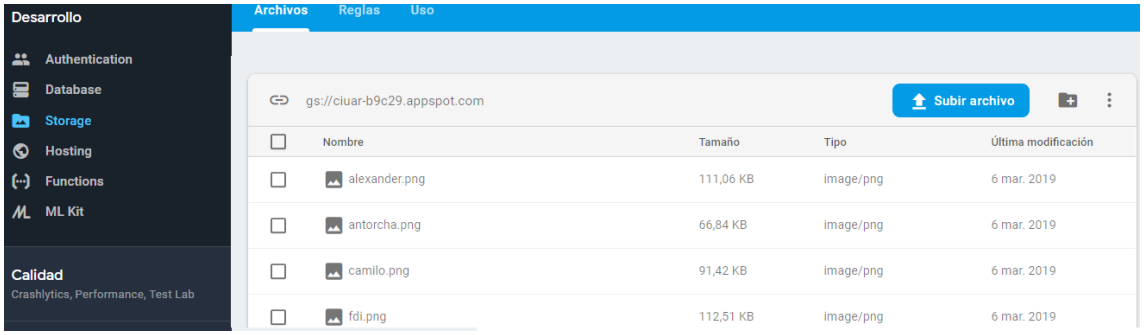

Figura 20. Storage de Firebase

4.5. Google Maps

En este punto se explicará el método seguido para usar el servicio de mapa de Google Maps en la aplicación. Este servicio ofrece una API de uso gratuito pero limitado para los desarrolladores. Hay que mencionar que desde 2018 la API de Google Maps pasó de ser totalmente gratuita a ofrecer un límite de uso. Superado este límite se procede a cobrar por megas empleados en accesos al servicio. Por esta razón se obliga a introducir una tarjeta de crédito a la hora de darse de alta para usar la API.

Para poder usar las API de Google se debe registrar en Google Cloud Platform accesible desde el siguiente enlace: *https://cloud.google.com/?hl=es*

4.5.1.Google Maps en CIU Tour

En primer lugar se asignaron las dependencias pertinentes en el archivo *build.gradle*. siendo la dependencia principal *com.google.android.gms:play-services-maps:16.1.0*. Una vez conseguidas las dependencias, se importaron las librerías de *com.google.android.gms.maps* y se desarrolló la actividad "*Map*". Estas librerías son:

- com.google.android.gms.maps.CameraUpdateFactory
- com.google.android.gms.maps.GoogleMap
- com.google.android.gms.maps.OnMapReadyCallback
- com.google.android.gms.maps.SupportMapFragment
- com.google.android.gms.maps.model.LatLng
- com.google.android.gms.maps.model.MarkerOptions

Esta actividad, al ser llamada, se encargará de contactar con la capa de modelo para acceder a los objetos *Locations* almacenados en Firebase. El objeto *Location* es una ubicación concreta de un monumento que dispone de su latitud, longitud y nombre.

Tras obtener una lista con todas las ubicaciones o *Locations*, se llama al método *displayMap* encargado de crear un objeto de tipo *GoogleMap* que representará el mapa de la ciudad. Se le asigna un punto del mapa para poder hacer zoom y centrar la vista (aquí en Ciudad Universitaria) y finalmente por cada uno de los objetos *Location* de la lista de objetos devuelta por la consulta a la base de datos, se añade un marcador al objeto *googleMap*. Este marcador consta de una variable *LatLng* con los atributos latitud y longitud de un monumento y atributo *title* que será el nombre de este.

Una vez añadidos todos los marcadores se realizará la animación de cámara o zoom para centrarse en Ciudad Universitaria, ya con todos los puntos de interés o monumentos mostrados.

El usuario podrá pulsar cualquier marcador para ver el nombre del monumento y tendrá la posibilidad de buscar el camino más corto hasta él.

4.6. Conclusiones

Como conclusión a esta sección, se consiguieron con éxito las metas propuestas para la realización de la aplicación. Tanto a nivel de código como planificación de la misma los resultados entran dentro del rango esperado. Algunas tecnologías requirieron mayor tiempo para su implementación que otra como en el caso de TensorFlow.

El uso de aplicaciones pertenecientes al mismo desarrollador (Google) facilitó mucho el trabajo a la hora de desarrollar la aplicación, pues permite la integración de las mismas con más facilidad, existiendo documentación a tal efecto distribuida por Google.

5. Manual de usuario

En esta sección se mostrarán las vistas más significativas de la aplicación con el fin de familiarizar al usuario con ellas mediante sus correspondientes descripciones y formas de uso.

La vista principal que se le presenta al usuario muestra un texto que le sugiere comenzar directamente con el reconocimiento de monumentos mediante imágenes. Para ello debe hacer clic en el botón de reconocer con forma de ojo (ver Figura 21). Tras tomar la foto y aceptarla, se mostrará la información del monumento que aparezca en ella. La información consta del nombre del autor, su fecha de construcción y última modificación y una breve descripción. Una vez en esta pantalla también se dispondrá de la opción de texto a audio mediante el botón inferior derecho con símbolo de altavoz (ver Figura 22).

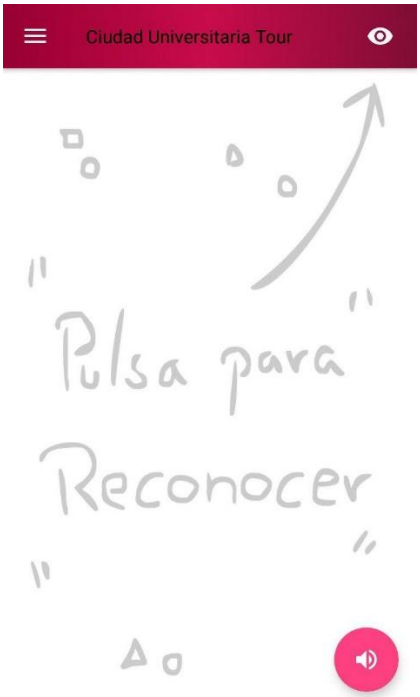

Figura 21. Vista principal de la aplicación

Figura 22. Vista de reconocimiento de Monumento

De no reconocerse la foto, se mostrará una vista de objeto desconocido (ver Figura 23). En ella también se puede acceder a la función de texto a audio.

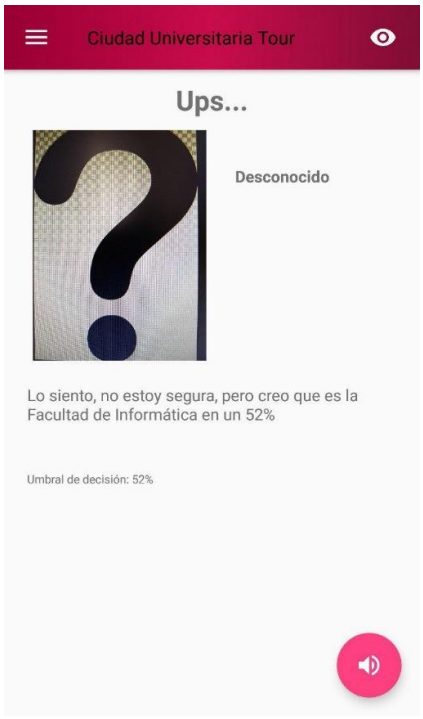

Figura 23. Vista de error de reconocimiento

Existe un menú para acceder a las diversas funcionalidades de la aplicación (ver Figura 24). Al menú se puede acceder con el botón de menú superior derecho. El menú contiene las siguientes opciones:

- Reconocer: Donde se accede a la misma funcionalidad del botón anterior de reconocer.

- Mapa: Se accede al mapa de Ciudad Universitaria donde se muestran las ubicaciones de los monumentos con su nombre y la posibilidad de encontrar la ruta más rápida hasta ellos (ver Figura 25).

- Monumentos: Se muestra una lista de los monumentos que se pueden ver en un recorrido en forma de imágenes con sus respectivos títulos (ver Figura 26).

- Historia: La vista que se muestra es una lista de hipervínculos que redirigen a las páginas oficiales de la UCM con la información de su historia (ver Figura 27).

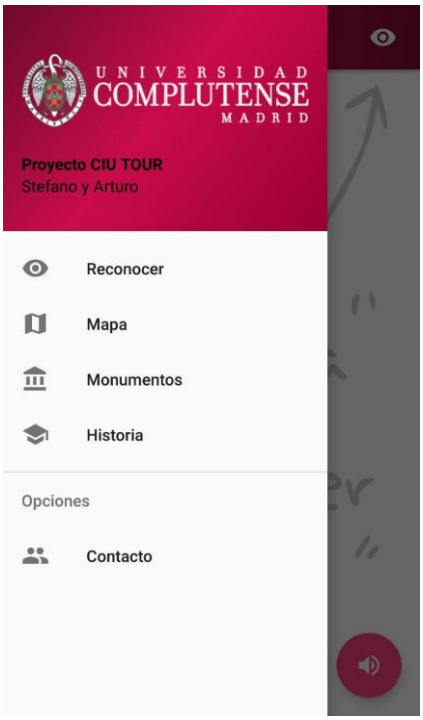

Figura 24. Menú de la aplicación

Figura 25. Mapa de Ciudad Universitaria

Figura 26. Lista de monumentos

Figura 27. Historia

Como puede observarse, la aplicación cumple el criterio de sencillez buscado durante la fase de diseño, tanto visualmente con un aspecto sencillo, amigable y elegante, como funcionalmente, con el desarrollo completo de las distintas funcionalidades que ofrece la aplicación.

6. Evaluación general

La aplicación se ha desarrollado en su totalidad y se ha publicado en Play Store, plataforma de distribución digital para dispositivos con sistema operativo Android. Después, se ha procedido a realizar dos evaluaciones para la aplicación: un análisis por parte de usuarios sobre la calidad del producto (sección 6.1), y otro análisis para identificar la efectividad en el reconocimiento y clasificación de edificios y monumentos (sección 6.2).

6.1. Evaluación de los usuarios

En esta sección se mostrarán los resultados de la evaluación de la aplicación por parte de usuarios reales. Recordamos que a estos se les ha pedido que descarguen, desde la plataforma de Google Play, y utilicen la aplicación y la evalúen a través de una encuesta online. En ningún momento se les ha indicado como usarla, solo los edificios y monumentos que el sistema permite reconocer.

6.1.1. Objetivos de la investigación

La encuesta se realizó para evaluar los siguientes aspectos:

1- Funcionalidad de la aplicación (fiabilidad del reconocimiento de imágenes, ayuda auditiva y localización adecuada en el mapa de los monumentos).
2- Utilidad de la aplicación (sea o no miembro de la UCM).
3- Comodidad en el uso de la aplicación.
4- Innovación.
5- Tipo de usuario.
6- Satisfacción.
7- Mejoras futuras.

El formulario usado para recopilar estos datos ha constado de las siguientes preguntas:

- Rango de edad (15-20 años, 20-25 años, 25-30 años, 30 años o más)
- ¿Qué tipo de aplicaciones tienes actualmente en tu dispositivo? (Apps sociales (facebook, twitter, instagram...), Apps de viajes, Apps de juegos, Apps de entretenimiento (películas, libros, deportes, ...), Apps de noticias, Apps de utilidad (calculadora, traductor, ...), Apps de compras).

Opinión del usuario (Valoración de 1 a 5, siendo 1 Totalmente en desacuerdo y 5 totalmente de acuerdo).

- La descarga de la aplicación ha sido fácil y sencilla.
- ¿Cómo ha sido tu primera reacción al entrar en la app?
- Localiza adecuadamente los monumentos en el mapa.
- La ayuda auditiva de lectura corresponde al texto mostrado en pantalla.
- Reconoce adecuadamente los objetos enfocados.
- Me gustaría que se desarrollara completamente la app.
- Creo que la aplicación es útil.

- Creo que la aplicación es fácil y cómoda de usar.
- Creo que la aplicación funciona correctamente.
- Creo que la aplicación podría ser interesante para estudiantes de la UCM.
- Creo que la aplicación podría ser interesante para el público en general.
- Creo que la app es innovadora.
- Creo que recomendaría esta app a un amigo o familiar.
- En general, estoy satisfecho con la app.

Expresa tus opiniones o recomendaciones sobre la app

- ¿Qué es lo que más te ha gustado de la app?
- ¿Qué es lo que menos te ha gustado de la app?
- ¿Qué mejoras futuras te gustaría que se desarrollasen para la aplicación?

6.1.2.Resultados

Los siguientes resultados han derivado de 35 usuarios encuestados de más de 15 años. El rango de edad puede verse en la Tabla 4.

Tabla 4. Tabla de rango de edad

	15 a 20 años	20 a 25 años	25 a 30 años	30 años o más
Porcentaje (%)	2,9	14,3	37,1	45,7

En primer lugar, con respecto a la funcionalidad de la aplicación, se evaluaron diferentes aspectos.

Respecto al reconocimiento de objetos, el 88,6% de los usuarios encuestados afirman que funciona adecuadamente. El 11,4% restante lo puntúa con un 3 sobre 5, lo que hace pensar que pueden haber tenido alguna dificultad a la hora del reconocimiento. La evaluación de este punto (reconocimiento de objetos) se verá en la sección 5.2.

El 100% de los usuarios confirman que la ayuda auditiva de lectura se corresponde con el texto mostrado en pantalla.

Por último, en cuanto a la localización en el mapa de los monumentos, vemos que el 88,6% afirman que lo realiza adecuadamente. Sólo el 11,4% lo valoran en una puntuación de 3 sobre 5.

En segundo lugar, los usuarios han evaluado la utilidad de la aplicación. El 97,1% han afirmado que la aplicación es útil. Además, el 97,1 cree que la aplicación es de interés para los estudiantes de la UCM y el 91,4% cree que sería de interés para el público en general.

En tercer lugar, vemos que la comodidad en el uso de la aplicación ha sido evaluada positivamente por el 97,1% de los usuarios. Además, el 94,3% afirman que la descarga de la aplicación ha sido fácil y sencilla. Sólo el 5,7% parecen haber tenido alguna dificultad para hacerlo.

En cuanto a la innovación, el 94,2% de los usuarios han afirmado que la aplicación les parece innovadora.

Como información adicional, el 97,1% evalúan de manera positiva la primera impresión al abrir la aplicación. El 97,1% de los usuarios afirman que les gustaría que se desarrollase más la aplicación.

Por último, respecto a la satisfacción del usuario vemos que el 100% han quedado satisfecho con el uso de la aplicación.

Como aspectos positivos a destacar, los usuarios mencionan los siguientes:

- El reconocimiento de objetos es adecuado.
- La información acerca de los monumentos de la universidad.
- La facilidad de uso de la aplicación.
- La interfaz sencilla, intuitiva y estéticamente cuidada.
- La ayuda auditiva.

Como aspectos a mejorar de la aplicación los usuarios proponen los siguientes:

- Mejorar el reconocimiento de objetos. Esto implica optimizar el proceso de clasificación de los edificios y monumentos.
- Mejorar las descripciones de los monumentos. Esto implica realizar un estudio profundo sobre la historia de cada edificio o monumento que puede reconocerse en la aplicación. Puesto que el objetivo era que mostrase información básica, no se ha empleado mucho tiempo en el desarrollo de este aspecto.
- La interfaz no está completa. Introducir el menú en todos las vistas de la aplicación.
- Mayor desarrollo en general de aplicación.

Por último, como propuestas de mejora de la aplicación mencionan los siguientes:

- Existencia de una opción para acceder directamente a la cámara del móvil.
- Aumentar el número de monumentos y edificios que reconoce la aplicación.
- Mostrar información del reconocimiento en distintos idiomas.
- Introducir un sistema GPS que permita indicar la distancia desde la localización del usuario hasta los diferentes monumentos.

6.2. Evaluación y resultados de la precisión obtenida en el reconocimiento de objetos

Al realizarse el proceso de entrenamiento de un proceso de aprendizaje automático es necesario determinar un número mínimo de imágenes para que el entrenamiento sea mínimamente correcto. A mayor número de imágenes, mejor será la precisión en el modelo de clasificación. Sin embargo, no solo el número de imágenes determina la precisión. También influye si las imágenes son una buena representación de las categorías por las que se las clasificará posteriormente.

En este proceso, se corre el riesgo de sobreajustar (*overfitting*) o subajustar (*underfitting*) el modelo al generalizar el patrón que permite la clasificación de un objeto.

- *Underfiting*: si nuestros datos de entrenamiento son insuficientes, el modelo no es capaz de generalizar un patrón suficientemente o reconocerá todo lo que vea como la misma clase. Por ejemplo, que un sistema identifique a todos los animales con cuatro patas como un perro, lo sean o no.
- *Overfitting*: si los datos de entrenamiento son excesivos, el modelo asimilará una serie de características comunes, y en cuanto se le exija reconocer ese mismo objeto con una característica ligeramente distinta, no podrá reconocerlo por no cumplir exactamente con todas las características. Un ejemplo es el color o la luminosidad, que no tienen por qué mantenerse estables.

Es muy común caer en estos problemas. La manera para poder determinar si el sistema es preciso en la clasificación de una muestra, consiste en realizar una validación del modelo mediante la clasificación de muestras de validación que no han sido utilizadas en el entrenamiento del modelo y cuya categoría ya conocemos previamente.

Para ello, se ha realizado una prueba real mediante una serie de capturas de cada edificio o monumento, anotándose los resultados de la precisión devuelta por TensorFlow para cada imagen y haciendo la media.

Si el modelo tiene un porcentaje alto de precisión en las muestras usadas en el entrenamiento, y un porcentaje bajo en las muestras de validación, estamos ante un caso de *overfitting*. Si el modelo solo acierta una clase especifica o el único resultado que se obtiene es siempre el mismo, se puede determinar un problema por *underfitting*.

Conociendo el número de fotos que se han sacado de cada monumento o estatua y la media de la precisión obtenida, se ha realizado la evaluación. Estos datos se pueden ver en la Tabla 5.

Tabla 5. Tabla de precisión media obtenida en las pruebas de reconocimiento

Objeto	Nº imágenes usadas en entrenamiento	Nº pruebas realizadas	Valor medio de la precisión en la clasificación con muestras no usadas en el entrenamiento (%)	Valor medio de la precisión en la clasificación con muestras usadas en el entrenamiento (%)	Riesgo (overfitting/ underfitting)
Estatua de Alexander Dubcek	73	30	89.53	99.38	*overfitting*
Estatua de Los Portadores de la Antorcha	287	30	93.03	99.33	*overfitting*
Estatua de Camilo José Cela	170	30	95.93	98.01	*overfitting*
Estatua de Omar Jaymay	79	30	91	99.95	*overfitting*
Facultad de Informática	168	30	78.06	97.55	*overfitting*
Facultad de Geografía e Historia	215	30	95.83	99.02	*overfitting*
Edificio Multiusos	43	30	97.5	98.15	*overfitting*
Rectorado	38	30	85.11	97.07	*overfitting*

Podemos ver claramente que todos los modelos se han sobreajustado. En este caso, al tener un número de imágenes muy reducido, podemos concluir que el número de pasos usado durante el proceso de entrenamiento ha sido excesivo con respecto al número de imágenes.

Analizando el valor medio de la precisión en la clasificación de los distintos elementos en relación con el número de imágenes, podemos ver dos patrones:

- **Estatuas**: puede verse que, a pesar de la disparidad en el número de fotografías tomadas por monumento, el valor medio de precisión ronda el 90%. Esto tiene una explicación. Al tener una superficie pequeña, se han podido realizar un mayor entrenamiento sobre una misma zona, por lo que con menor entrenamiento se tiene una buena media.

- **Edificios**: puede verse que la diferencia en el número de fotos tomadas tiene un impacto importante en el valor de precisión media. Esto se debe a que al hacerse fotos desde todos los ángulos de un edificio y ser este muy grande, cada paso del entrenamiento debe abarcar una superficie muy amplia de un edificio y no siempre identifica una sección del edificio.

Hay una excepción a esta regla, el edificio multiusos. Tras revisar las fotos usadas en el entrenamiento, puede verse que casi todas las fotos se realizaron sobre la fachada del edificio, por lo que el entrenamiento se concentró sobre esa superficie, no sobre todo el edificio, y por eso con un número menor de fotos, se ha obtenido una buena media.

Estos resultados se han hecho con una variedad limitada de fotos y un proceso de ajuste poco optimizado, pues el objetivo era que se realizara el reconocimiento y no la optimización de dicho proceso. Para ello, se necesitaría una cantidad muy superior de imágenes de entrenamiento y un proceso de optimización del ajuste en el entrenamiento del modelo de clasificación. Esta tarea queda como parte del trabajo futuro especificado en la sección 6.2.

6.3. Conclusión

Como conclusión de esta sección, se pueden han obtenido los resultados de las dos evaluaciones, una con usuarios reales y otra de la precisión en la clasificación de los edificios y monumentos de la UCM.

Puede verse que la satisfacción de los usuarios con los diferentes aspectos de la aplicación es alta, a pesar de la detección de pequeños errores. También se puede determinar que el proceso de entrenamiento para el algoritmo de clasificación de las imágenes no ha sido optimizado, debido al exceso de entrenamiento sobre un número pequeño de imágenes de entrenamiento.

7. Conclusiones

En esta sección se realizarán las conclusiones, basadas en el cumplimiento de los objetivos determinados en la sección 1.1 y en los resultados de evaluación de la aplicación vistos en la sección 6. Así mismo, se realizará una sección de trabajo futuro, donde determinaremos las posibles iteraciones futuras para la aplicación.

7.1. Conclusiones

El objetivo general de este proyecto era el diseño e implementación de una aplicación móvil para visitas turísticas que permitiese al usuario obtener información sobre algunos edificios y monumentos mediante el aprendizaje automático y el reconocimiento de imágenes. Este objetivo general se ha cumplido.

La aplicación desarrollada permite al usuario reconocer los siguientes edificios y monumentos de la UCM al visitarlos:

- Estatua de Los Portadores de la Antorcha.
- Estatua de Alexander Dubcek.
- Estatua de Omar Jaymay.
- Estatua de Camilo José Cela.
- Facultad de Informática.
- Facultad de Geografía e Historia.
- Edificio Multiusos.
- Rectorado.

Al realizar una fotografía, la aplicación es capaz de identificar y mostrar información básica relacionada con el edificio o monumento fotografiado. En caso de no reconocer el edificio o monumento, lo indica. No se ha realizado un proceso de investigación exhaustivo de la información proporcionada para los distintos monumentos o edificios, pues no era necesario para cumplir el objetivo general. Por ello, solo se ha proporcionado información básica, con la intención de ampliarla como mejora futura en base a una documentación histórica y documentada.

Así mismo, se han cumplido cada uno de los objetivos específicos determinados en la sección 1.1. Para ello, se han realizado satisfactoriamente las siguientes tareas:

1- Se ha realizado un análisis de las diferentes aplicaciones relacionadas con el sector turístico y/o con el reconocimiento de imágenes existentes en el mercado actual buscando las características y criterios a favor y en contra de cada una. De esta forma, se establecieron ciertos criterios necesarios y deseables para la aplicación en la sección 2.3 en base a la usabilidad y comodidad de uso. Igualmente, se han definido otros criterios que se desarrollarían en un futuro al no ser necesarios para la realización del objetivo general, pero que podrían añadir funcionalidades útiles para el usuario en un futuro.

   Cabe destacar que el reconocimiento y clasificación de una imagen era un requisito imprescindible, así como hacer la interfaz y el proceso de reconocimiento lo más simple posible para el usuario. Así mismo, se determinó

que el uso de la realidad aumentada o virtual no se implementaría, pero que sería útil como desarrollo futuro para mejorar la interactividad del usuario con la aplicación.

2- Se ha realizado un análisis sobre las tecnologías y herramientas de *machine learning* disponibles en el momento del desarrollo de la aplicación. De entre todas ellas, se ha elegido y aplicado la más adecuada. En este caso TensorFlow, que ofrece numerosas funcionalidades, el uso de una tecnología enfocada a crear aplicaciones móviles ligeras y una gran cantidad de documentación, así como el apoyo de una amplia comunidad de desarrolladores.

3- Se ha realizado un entrenamiento con un número variable de imágenes de muestra de edificios y monumentos de la UCM mediante TensorFlow. Para ello hubo que tomar individualmente fotografías de algunos de los edificios y monumentos localizados en la UCM y realizar un proceso de entrenamiento con el que se generó un grafo computacional que incorporaríamos a la aplicación desarrollada con posterioridad. Tras realizar los análisis de la precisión de este proceso, se detectaron deficiencias en el proceso de optimización de la clasificación y se determinaron los motivos. Entre estos se incluyen el limitado número de imágenes utilizados en el entrenamiento y el sobreajuste derivado de un exceso de entrenamiento en el proceso de la generación del grafo computacional.

4- Se ha diseñado y realizado satisfactoriamente la aplicación, mediante un proceso incremental e iterativo, realizándose distintas versiones de la aplicación, incorporando o modificando distintos aspectos hasta obtener una aplicación que cumpliese el objetivo general. Se ha procurado mantener una interfaz cómoda para el usuario, así como incorporar elementos para mejorar la experiencia del usuario y la usabilidad de la aplicación. Se ha implementado el uso de la geolocalización y también el uso de la tecnología para convertir el texto en voz.

5- Se han realizado dos evaluaciones. Una con usuarios reales y otra de precisión en la clasificación de los edificios y monumentos de la UCM. Los resultados de ambos análisis pueden verse en las secciones 6.1.2 y 6.2.2 del documento. Puede verse que la satisfacción de los usuarios con los diferentes aspectos de la aplicación es muy alta, a pesar de la detección de algunos errores. También se puede determinar que el proceso de entrenamiento para el algoritmo de clasificación de las imágenes no ha sido optimizado, debido al exceso de entrenamiento sobre pocas imágenes de entrenamiento.

Con los objetivos cumplidos y los análisis hechos, se ha concluido que la aplicación ha satisfecho en general a los usuarios, a pesar de posibles correcciones o mejoras que puedan hacerse de la aplicación.

7.2. Trabajo futuro

Gracias al proceso de evaluación realizado, se han determinado algunos fallos de diseño en la interfaz y posibles mejoras en cuanto a la usabilidad, la funcionalidad y la personalización de la aplicación, que incluyen incorporación de la realidad aumentada o la realidad virtual, la ampliación de la base de datos, mejoras en la interfaz y la

optimización del proceso de clasificación, todo ello detallado en la sección 7.2 de mejoras futuras. Cabe destacar que el proceso de ampliación de las imágenes de entrenamiento y la optimización del proceso de clasificación, serían prioritarias dentro de dichas mejoras. Estas mejoras futuras para la aplicación se explican con más detalle a continuación.

7.2.1. Optimización del proceso de entrenamiento de clasificación

Hemos visto que el proceso de clasificación de una imagen no es óptimo. Para mejorar el proceso de clasificación de una imagen, se deberán implementar las siguientes medidas:

- **Uso de las herramientas de optimización ofrecidas por TensorFlow**. La comunidad de TensorFlow ofrece ciertas herramientas desarrolladas para realizar un proceso de optimización sobre el grafo generado, para mejorar su eficiencia y la velocidad necesaria para realizar la clasificación. TensorFlow provee también de una herramienta de visualización del proceso de entrenamiento, para facilitar la comprensión, depuración y optimización de este proceso, permitiendo la generación de gráficos con información del rendimiento.

- **Uso de la mínima cantidad de imágenes de entrenamiento**. Para evitar el sobreajuste del sistema, será necesario determinar el mínimo conjunto de imágenes necesario para el correcto proceso de entrenamiento, tanto para entrenar el modelo como para validarlo. Un número mayor provocará que el sistema se sobreajuste, asimilando ciertas características que impedirán la clasificación de las imágenes con elementos anómalos, como personas delante del objeto a reconocer o cambios en la luz.

- **Imágenes variadas y equilibradas en cantidad**. Para evitar el *underfitting*, el sistema precisará de una muestra de imágenes amplia y variada de los elementos de cada clase, que permita al sistema disponer de suficientes criterios para realizar una clasificación adecuada. Para evitar el *overfitting* de una clase con respecto a las demás, procuraremos realizar el entrenamiento con un número similar de imágenes para cada clase.

- **Conjuntos de imágenes para el entrenamiento y validación separados**. Será necesaria la división del conjunto total de imágenes obtenidas de cada clase en dos subconjuntos. Uno para el entrenamiento y otro para la validación. El conjunto de validación, aunque menor, deberá disponer de una amplia variedad de muestras para comprobar los resultados una vez entrenado el modelo. Esto nos permitirá realizar pruebas sobre el proceso de ajuste de la precisión de la clasificación de manera inmediata a la generación del grafo computacional generado, para poder realizar las evaluaciones de forma sencilla.

7.2.2. Mejora de la interfaz

Con una mejora de la interfaz se logrará mayor cercanía con los usuarios y mejor movilidad dentro de la aplicación. Los puntos importantes por reforzar en este tema serían los siguientes:

Bloqueo y desbloqueo de monumentos. Se añadirá un sistema o juego por el cual los monumentos se muestren bloqueados y para acceder a ellos o desbloquearlos sea necesario haberlos reconocido antes. Esto motivará a los usuarios a visitar todos los monumentos.

Visualización amigable de la lista de monumentos: La lista de monumentos podría presentar una vista más amigable. Por ejemplo, con sistemas de presentación de imágenes por carrusel o en cascada, que redirigirán al usuario a una ventana con la información de cada uno de ellos (si no están bloqueados).

Mejoras en la geolocalización y el mapa: El mapa se podrá mejorar añadiendo funcionalidad, como, por ejemplo:

- Geolocalización incluida en el proceso de clasificación para mejorar asegurando una precisión mayor a la hora de mostrar la información.

- Distancia con respecto al monumento. Una mejora sugerida por los usuarios es poder determinar la distancia al edificio o monumento visualizado, para poder hacerse una idea del tamaño.

- Mejora en la interfaz de la etiqueta de geolocalización, añadiendo botones más llamativos al acceso a las rutas hacia los monumentos, así como cierta información básica sobre el edificio o monumento, como por ejemplo una fotografía.

- **Introducción de un área personal**: Añadir un área personal en la cual los usuarios puedan ingresar y registrar los monumentos visitados. También proporcional nombre de avatar e imagen de perfil, así como opciones de dejar comentarios en cada monumento reconocido o indicar sus favoritos.

- **Evitar el acceso a contenido externo a la aplicación**. Para evitar que los usuarios accedan a contenido externo a la aplicación, toda la información necesaria sería provista por la propia aplicación y no mediante links a URLs externas como el caso del apartado de Historia. Además, la información podrá ser presentada de forma más personalizada para el usuario.

7.2.3. Ampliación de la información contenida en la base de datos

Uno de los aspectos que se ha visto necesario tras el análisis de la aplicación por parte de usuarios es la ampliación de la información ofrecida, tanto del número de elementos que es posible identificar, como la información mostrada de cada uno de ellos. Para ello, se considerará el uso de dos estrategias:

- Colaboración con la UCM para la adquisición de información sobre los edificios y monumentos localizados en sus instalaciones, como por ejemplo imágenes, datos históricos relevantes, curiosidades y otros elementos de interés turístico.
- Ampliación del contenido con la colaboración de los usuarios. Se permitirá a los usuarios disponer de una herramienta que permita incluir fotografías de

monumentos o edificios ya existentes en el sistema o incluso añadir nuevos monumentos. Estas nuevas imágenes se gestionarán por un servicio web, que enviará dichas imágenes a un repositorio para hacer nuevas clasificaciones.

7.2.4. Redes sociales

Se añadirá una conexión a distintas redes sociales para mejorar la publicidad y expansión de la aplicación, permitiendo compartir las experiencias de los usuarios, los recorridos realizados y los puntos de interés que más les han gustado. Así mismo, los comentarios hechos en la red social permitirán conocer la opinión de los usuarios respecto de la aplicación y, gracias a ello, poder detectar mejoras futuras. Además, se podrá incluir notificaciones de eventos o nuevas características.

7.2.5. Realidad aumentada

"El término realidad aumentada (RA) hace referencia a la visualización directa o indirecta de elementos del mundo real combinados (o aumentados) con elementos virtuales generados por un ordenador, cuya fusión da lugar a una realidad mixta. El aumento de la realidad tiene lugar en tiempo real y se produce en consonancia semántica con objetos del entorno." (Cobo & Moravec, 2011)

En general, la realidad aumentada debe cumplir las siguientes propiedades:

- Combinación de objetos reales y virtuales en ambientes integrados.
- Las señales y su reconstrucción se ejecutan en tiempo real.
- Alineación de objetos reales y virtuales entre sí.
- Los objetos reales y virtuales son registrados y alineados geométricamente entre ellos y dentro del espacio, para darles coherencia espacial.

Esta tecnología se encuentra ya en muchos ámbitos laborales como pueden ser diseño de interiores o el turismo. Un ejemplo muy claro del potencial de AR es la aplicación de IKEA para diseñar una habitación a medida insertando modelos 3D en un entorno real.

Figura 28. Ejemplo de IKEA con AR

Añadir realidad aumentada podrá ayudar a los usuarios a tener una experiencia más inmersiva a la hora de usar la aplicación, proporcionando información visual a la vez que se visualiza el objeto real.

1. *Guía turístico*

La implementación de un guía turístico modelado en 3D y animado aparecerá cerca de cada monumento al realizarse el reconocimiento de un edificio o un monumento. Este guía contará la historia del monumento, guiará al usuario hasta el resto de los monumentos, ofrecerá consejos de rutas, etc. Esto proporciona un entorno más interactivo y atractivo para el usuario.

2. *Reconstruir Ciudad Universitaria*

El uso de la realidad aumentada para simular reconstrucciones de la Ciudad Universitaria en sus diferentes épocas históricas permitirá al usuario realizar un seguimiento de la historia a través de una experiencia mucho más inmersiva. Al enfocar ciertos puntos, sean edificios, monumentos o carteles, el usuario podrá visualizar una reconstrucción 3D de cómo era este espacio en otra época.

Este desarrollo sería uno de los más costosos debido al proceso de estudio histórico, el diseño y reconstrucción de los modelos 3D, así como el trabajo de integración de dichos modelos en el entorno real. Necesitaría incluir además en el equipo diseñadores gráficos, historiadores y otros perfiles.

8. Conclusions and future work

In this section there will be a section of conclusions, based on the fulfillment of the objectives determined in section 1.1 and on the results of evaluation of the application as seen in section 6. Likewise, there will be a section of future work, where we will determine the possible future iterations for the application.

8.1. Conclusions

The objective of this project was the design and implementation of a mobile application for tourist visits that would allow the user to obtain information about some buildings and monuments through automatic learning and image recognition. This objective has been met.

The developed application allows the user to recognize the following buildings and monuments of the UCM when visiting them:

- Estatua de Los Portadores de la Antorcha.
- Estatua de Alexander Dubcek.
- Estatua de Omar Jaymay.
- Estatua de Camilo José Cela.
- Facultad de Informática.
- Facultad de Geografía e Historia.
- Edificio Multiusos.
- Rectorado.

When taking a photograph, the application is able to identify and display basic information related to the building or monument photographed. In case of not recognizing the building or monument, it indicates it. There has not been a process of exhaustive investigation of the information provided for the different monuments or buildings, as it was not necessary to meet the general objective. For this reason, only basic information has been provided, with the intention of extending it as a future improvement on the basis of historical and documented documentation.

In addition, each of the specific objectives set out in section 1.1 has been met. To this end, the following tasks have been satisfactorily carried out:

1. An analysis of the different applications related to the tourism sector and/or the recognition of existing images in the current market has been carried out, looking for the characteristics and criteria for and against each one. In this way, certain necessary and desirable criteria were established for the application in section 2.3 based on usability and ease of use. Likewise, other criteria have been defined that would be developed in the future as they are not necessary for the achievement of the general objective but could add useful functionalities for the user in the future.

   It should be noted that the recognition and classification of an image was a prerequisite, as well as making the interface and recognition process as simple as possible for the user. Likewise, it was determined that the use of augmented or

virtual reality would not be implemented, but that it would be useful as a future development to improve the interactivity of the user with the application.

2. An analysis has been carried out on the machine learning technologies and tools available at the time of application development. From all of them, the most suitable one has been chosen and applied. In this case TensorFlow, which offers numerous functionalities, the use of a technology focused on creating light mobile applications and a large amount of documentation, as well as the support of a large community of developers.

3. A training has been carried out with a variable number of sample images of buildings and monuments of the UCM by means of TensorFlow. In order to do this, it was necessary to take individual photographs of some of the buildings and monuments located in the UCM and carry out a training process with which a computational graph was generated that we would incorporate to the application developed later. After analyzing the precision of this process, deficiencies were detected in the process of optimization of the classification and the reasons were determined, including the limited number of images used in the training and the over-adjustment derived from an excess of training in the process of generating the computational graph.

4. The application has been designed and carried out satisfactorily, by means of an incremental and iterative process, carrying out different versions of the application, incorporating or modifying different aspects until obtaining an application that fulfils the general objective. We have tried to maintain a comfortable interface for the user, as well as to incorporate elements to improve the user experience and the usability of the application. The use of geolocation has been implemented, as well as the use of technology to convert text into voice.

5. Two evaluations have been carried out. One with real users and another of precision in the classification of the buildings and monuments of the UCM. The results of both analyses can be seen in section 5.3 of the document. User satisfaction can be seen despite error detection. It can also be determined that the training process for the image classification algorithm has not been optimized, due to the excess of training over few training images.

With the objectives met and the analyses made, it has been concluded that the application has generally satisfied users, despite possible corrections or improvements that can be made to the application.

Thanks to the evaluation process, several design flaws in the interface and possible improvements in terms of usability, functionality and customization of the application have been identified, including the incorporation of augmented reality or virtual reality, the expansion of the database, improvements in the interface and the optimization of the classification process, all detailed in section 7.2 of future improvements. It should be noted that the process of enlarging the training images and optimizing the classification process would be a priority within these improvements.

8.2. Future work

Following the assessment made and considering the responses of users with regard to future improvements, the decision has been taken to undertake the following future improvements to the implementation:

8.2.1. Optimization of the classification training process

We have seen that the process of classifying an image is not optimal. In order to improve the image classification process, the following measures should be implemented:

- **Use of the optimization tools offered by TensorFlow**. The TensorFlow community offers certain tools developed to carry out an optimization process on the generated graphic, to improve its efficiency and the necessary speed to carry out the classification. TensorFlow also provides a visualization tool of the training process, to facilitate the understanding, debugging and optimization of this process, allowing the generation of graphs with performance information.

- **Use of the minimum amount of training images**. To avoid system over-adjustment, it will be necessary to determine the minimum set of images necessary for the correct training process, both to train the model and to validate it. A greater number will cause the system to over-adjust, assimilating certain characteristics that will prevent the classification of images with anomalous elements, such as people in front of the object to recognize or changes in the light.

- **Images varied and balanced in quantity**. To avoid underfitting, the system will need a wide and varied sample of images of the elements of each class, which allows the system to have enough criteria to make a proper classification. To avoid overfitting one class with respect to the others, we will try to carry out the training with a similar number of images for each class.

- **Separate sets of images for training and validation**. It will be necessary to divide the total set of images obtained from each class into two subsets. One for training and one for validation. The validation set, although smaller, should have a wide variety of samples to check the results once the model has been trained. This will allow us to perform tests on the process of adjusting the accuracy of the classification immediately to the generation of the generated computational graph, to be able to make the evaluations in a simple way.

8.2.2. Interface Improvement

An improved interface will achieve greater closeness to users and better mobility within the application. The important points to reinforce in this topic would be the following:

- **Blocking and unblocking of monuments**. A system or game will be added by which the monuments are shown blocked and in order to access or unblock them it is necessary to have recognized them before. This will motivate users to visit all monuments.

- **Friendly display of the list of monuments**: The list of monuments could present a more user-friendly view, for example with carousel or cascade presentation systems, which will redirect the user to a window with the information of each of them (if they are not blocked).

- **Improved geolocation and mapping**: The map can be improved by adding functionality, for example:

    - Geolocation included in the classification process to improve ensuring greater accuracy when displaying information.

    - Distance from the monument. An improvement suggested by users is to be able to determine the distance to the building or monument displayed, to get an idea of the size.

    - Improvement in the interface of the geolocation label, adding more eye-catching buttons to the access of the routes to the monuments, as well as some basic information about the building or monument, such as a photograph.

- **Introduction of a personal area**: Add a personal area in which users can enter and register the monuments visited. Also provide avatar name and profile image, as well as options to leave comments on each recognized monument or indicate your favorites.

- **Avoid external links to the application**. To avoid the use of third party applications and improve usability, all necessary information would be provided by the application itself and not through links to external URLs as in the case of the History section. In addition, the information may be presented in a more personalized way for the user.

8.2.3. Extension of the information contained in the database

One of the aspects that has become necessary after the analysis of the application by users is the extension of the information offered, both in terms of the number of elements that can be identified, and the information shown for each of them.

To this end, the use of two strategies will be considered:

- Collaboration with the UCM for the acquisition of information on the buildings and monuments located in its facilities, such as images, relevant historical data, curiosities and other elements of tourist interest.

- Expansion of the content with the collaboration of users. Users will be provided with a tool to include photographs of existing monuments or buildings in the system or even to add new monuments. These new images will be managed by a web service, which will send these images to a repository with which new classifications can be made.

8.2.4. Social media

A connection to different social networks will be added to improve the advertising and expansion of the application, allowing users to share their experiences, the routes taken and the points of interest they liked the most. Likewise, the comments made on the social network will make it possible to know the opinion of users regarding the application and, thanks to this, be able to detect future improvements. In addition, you can include notifications of events or new features.

8.2.5. Augmented reality

"The term augmented reality (AR) refers to the direct or indirect visualization of real-world elements combined (or augmented) with virtual elements generated by a computer, the fusion of which gives rise to a mixed reality. The augmentation of reality takes place in real time and takes place in semantic consonance with objects in the environment". (Cobo & Moravec, 2011)

In general, augmented reality must meet the following properties:

- Combination of real and virtual objects in integrated environments.
- Signals and their reconstruction are executed in real time.
- Alignment of real and virtual objects with each other.
- Real and virtual objects are recorded and geometrically aligned with each other and within space to give them spatial coherence.

This technology is already found in many working environments such as interior design or tourism. A very clear example of AR's potential is IKEA's application to design a custom room by inserting 3D models in a real environment.

Adding augmented reality can help users to have a more immersive experience when using the application, providing visual information while visualizing the real object.

6. Tourist guide

The implementation of a 3D modeled, and animated tour guide will appear near each monument when a building or monument is surveyed. This guide will tell the history of the monument, guide the user to the rest of the monuments, offer route advice, etc. This provides a more interactive and attractive environment for the user.

7. Rebuild University City

The use of augmented reality to simulate reconstructions of the University City in its different historical periods will allow the user to follow history through a much more immersive experience.

By focusing on certain points, be they buildings, monuments or posters, the user will be able to visualize a 3D reconstruction of what this space was like in another era. This development would be one of the most expensive due to the historical study process, the design and reconstruction of 3D models, as well as the work of integrating these models in the real environment.

9. Contribuciones al proyecto

En esta sección se van a detallar las tareas que han realizado cada uno de los integrantes del grupo en las distintas fases para el desarrollo de este proyecto. Se debe tener en cuenta que la gran mayoría del trabajo realizado ha sido o bien realizado por ambos miembros de forma conjunta o revisado y corregido por ambos miembros.

9.1. Contribución de Arturo Marino Quintana

En este punto se redacta la contribución de Arturo Marino Quintana en el proyecto.

9.1.1. Estado del arte

El estado del arte fue un punto decisivo para la forma en la que se desarrollaría el proyecto, por lo que se dedicó mucho tiempo a indagar las diferentes tecnologías disponibles y a realizar el estudio de las características que debía tener la aplicación.

- Estudio de las características y funcionalidades de aplicaciones existentes en el mercado que realicen tareas de reconocimiento de imágenes, aplicaciones galardonadas en el ámbito del turismo y aplicaciones relacionadas con la realidad aumentada o la realidad virtual.

- Realización del análisis de la competencia.

- Recopilación de características sobre las herramientas de aprendizaje automático, así como de su funcionalidad e implementación. Se realizaron pruebas sobre la implementación de varias herramientas durante este proceso.

- Realización del análisis de las herramientas de aprendizaje automático.

- Recopilación de herramientas para la implementación de la base de datos, así como su uso e implementación.

9.1.2. Diseño e implementación

- Obtención de las imágenes para la realización del entrenamiento del grafo computacional usado para la clasificación de las imágenes. Para ello, junto son Stefano, se produjo a la captura de fotografías de los distintos elementos a identificar.

- Generación de grafo computacional usado para la clasificación de imágenes en TensorFlow. Este grafo hubo de generarse repetidas veces a medida que se añadían imágenes.

- Implementación de la vista principal, el menú y la vista de historia.

- Implementación, junto con Stefano, de TensorFlow en el proyecto.

- Diseño del Modelo Vista Controlador en conjunto con Stefano.

9.1.3.Evaluación con usuarios

La evaluación de los usuarios ha sido clave en la obtención de las conclusiones y para determinar el trabajo futuro de la aplicación, así como una medida autocrítica sobre el trabajo realizado, teniendo en cuenta la complejidad de las tecnologías con las que se ha tratado.

- Creación del formulario para la obtención de los criterios de evaluación junto con Stefano.

- Análisis de los resultados de la evaluación, en función de los resultados obtenidos a través del formulario previamente mencionado y a las pruebas realizadas y descritas en la sección 6.2 de la memoria.

- Conclusiones de las distintas evaluaciones.

9.1.4.Memoria

- Redacción del resumen de la memoria.

- Redacción de la introducción (sección 1), donde se incluyen la especificación de los objetivos, el plan de trabajo y una breve introducción al aprendizaje automático.

- Redacción del estado del arte (sección 2), incluyendo la recopilación de los datos obtenidos en la recopilación de la información sobre las distintas aplicaciones, herramientas y tecnologías y los análisis, así como las conclusiones.

- Redacción de evaluación general (sección 6), incluyendo el proceso para la obtención de los criterios de evaluación y el análisis de los resultados obtenidos.

- Redacción de las conclusiones (sección 7.1).

- Revisión y corrección general de la memoria.

9.2. Contribución de Stefano Mazzuka Cassani

En este punto se redacta la contribución de Stefano Mazzuka Cassani en el proyecto.

9.2.1.Estado del arte

- Estudio de las características y funcionalidades de aplicaciones existentes en el mercado que realicen tareas de reconocimiento de imágenes y aplicaciones

galardonadas en el ámbito del turismo. Hubo que buscar realizar un estudio de las aplicaciones semejantes con el propósito de añadir un factor diferencial a la hora de desarrollar CIU Tour.

- Recopilación de herramientas de aprendizaje automático, así como su funcionalidad e implementación. Se buscaron varias alternativas de aprendizaje automático e incluso otras como Vuforia se tuvieron en cuenta para realizar el reconocimiento de monumentos, pero finalmente se optó por TensorFlow.

- Recopilación de herramientas para la implementación de la base de datos, asi como su uso e implementación. De entre las herramientas conocidas e investigadas, se empleó Firebase como sistema de almacenamiento de datos de la aplicación:

- Análisis de las herramientas de base de datos.

9.2.2.Diseño e implementación

- Obtención de las imágenes de entrenamiento del grafo computacional. Se obtuvieron imágenes para cada monumento y edificio que se quería identificar.

- Generación de grafo computacional usado para la clasificación de imágenes en TensorFlow. El entrenamiento del grafo con las imágenes recolectadas tuvo un período de duración de cuarenta y cinco minutos.

- Creación de vistas: Lista de monumentos y mapa. La vista de mapa hace uso de la API de Google llamada Google Maps y la integración de la misma y el alta en la cuenta de Google APIs corrió al cargo de Stefano.

- Diseño del Modelo Vista Controlado. El diseño del modelo vista controlador corrió a cargo de ambos integrantes del equipo.

- Desarrollo de los diagramas: Creación de diagramas de clase y actividad.

- Diseño de la estructura de la BBDD. Se estructuró la base de datos según las necesidades de la aplicación.

9.2.3.Evaluación con usuarios

- Creación del formulario para la obtención de los criterios de evaluación junto con Arturo. Los formularios tenían como finalidad recoger información de usuarios que hubiesen empleado la aplicación para poder posteriormente tratar esos datos.

- Obtención del valor medio de la precisión en la clasificación con muestras no usadas en el entrenamiento (%) y valor medio de la precisión en la clasificación con muestras usadas en el entrenamiento (%), usados para la evaluación de la precisión obtenida en el reconocimiento y clasificación de edificios y monumentos.

Para esto se realizaron pruebas de reconocimiento tanto a edificios, empleando la aplicación terminada, como al propio grafo desde consola pasándole treinta imágenes de cada monumento y edificio.

9.2.4.Memoria

- Redacción de la sección 3 y 4, con el desarrollo de los sistemas relacionados y la implementación del proyecto.

- Manual de usuario (sección 5): Manual de instrucciones de uso de la aplicación para usuarios nuevos en CIU Tour.

- Redacción del trabajo futuro (sección 7.2).

- Creación del apéndice sobre la instalación de TersorFlow y la obtención del grafo computacional para la clasificación de imágenes (sección 11), dónde se recoge paso a paso el protocolo seguido para una correcta instalación del TensorFlow.

- Revisión general y correcciones sobre la memoria.

10. Bibliografía

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., & MacIntyre, B. (2001). Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications*, 14.

Cambronero, C. G., & Moreno, I. G. (2006). Algoritmos de aprendizaje: knn & kmeans. *Intelgencia en Redes de Comunicación, Universidad Carlos III de Madrid*. Obtenido de http://www.it.uc3m.es/~jvillena/irc/practicas/08-09/06.pdf

González, Y. D., & Romero, Y. F. (2012). Patrón Modelo-Vista-Controlador. Revista Telemática, 11(1), 47-57.

Kofler, M. (2001). What Is MySQL?. In MySQL (pp. 3-19). Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-0853-2_1

Minsky, M. (1986). La sociedad de la mente: la inteligencia humana a la luz de la inteligencia artificial. *Ediciones Galápago*.

Pajares, M. G., & De La Cruz, J. (2010). Aprendizaje Automático un Enfoque Práctico. *RA-MA SA Editorial y Publicaciones. ISBN-10: 8499640117*.

Pajares, G., & Santos, M. (2006). Inteligencia artificial e ingeniería del conocimiento. *Alfaomega Grupo Editor*.

Romaní, C. C., & Moravec, J. W. (2011). Aprendizaje invisible: Hacia una nueva ecología de la educación (Vol. 3). *Edicions Universitat Barcelona.*

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press. Obtenido de https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf

Sauter, P., Vögler, G., Specht, G., & Flor, T. (2005). A Model–View–Controller Extension for Pervasive Multi-client User Interfaces. Personal Ubiquitous Comput., 9(2), 100–107. https://doi.org/10.1007/s00779-004-0314-7

Schwaber, K. (1997). Scrum development process. In Business object design and implementation (pp. 117-134). Springer, London.

Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum (Vol. 1). Upper Saddle River: Prentice Hall.

Schwaber, K & Sutherland, J. (2017). La guía de Scrum. Recuperado de https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-SouthAmerican.pdf

Takeuchi, H., & Nonaka, I. (1986). The new new product development game. Harvard business review, 64(1), 137-146.

Tomás, D., Vicedo, J. L., Suárez Cueto, A., Bisbal Asensi, E., & Moreno Boronat, L. (2005). Una aproximación multilingüe a la clasificación de preguntas basada en aprendizaje automático. *Procesamiento del lenguaje natural, nº 35 (sept. 2005); pp. 391-398.*

11. Apéndices

En esta sección se detallará parte del trabajo extra realizado indicando paso a paso la correcta instalación de TensorFlow que se ha realizado durante la ejecución del proyecto. Se ha considerado la redacción de esta sección debido a la complejidad de la instalación y a los problemas de incompatibilidad entre versiones distintas de los distintos *software* involucrados.

11.1. Instalación TensorFlow

TensoFlow emplea scripts en Python, por lo cual el primer paso será el de instalación de Python 3.6.0. Es imperativo conservar las versiones que se expondrán para evitar futuros errores de compatibilidad entre TensoFlow, Python y otras herramientas.

- PASO I Instalar Python

Para la descarga se irá a la página oficial de Python que se encuentra en el siguiente enlace. Una vez ahí en el apartado de **Downloads** se hará click en **all releases** y se descargará la versión 3.6.0 como se muestra en la imagen 3.1

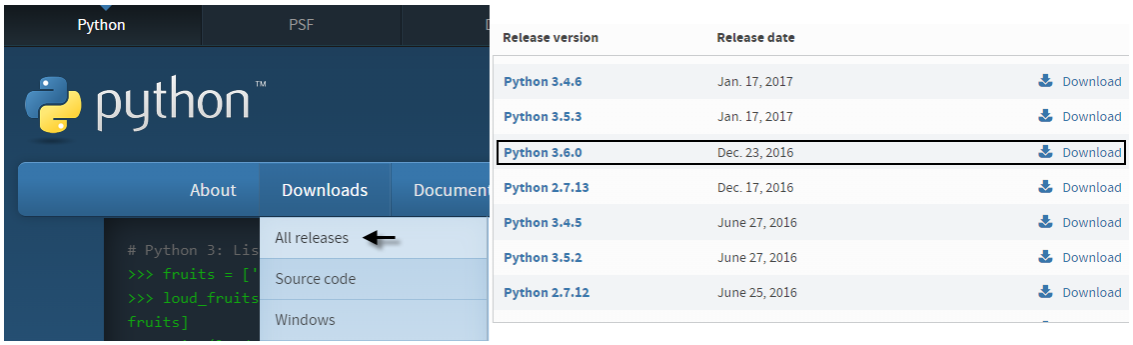

Imagen 3.1

Una vez descargado el archivo se procederá a la instalación del mismo. Muy importante es tener activo el check de agregar ruta al PATH para poder acceder a Python desde cualquier punto del sistema dentro de la consola de comandos.

Trans la instalación de Python es importante reiniciar el ordenador para posteriormente poder comprobar que ha sido exsitosa. Para esta comprobación se abrirá la consola de comandos y se introducirá la siguiente línea de comandos: `python --version` lo cual mostrará la versión de Python instalada en el sistema tal como se muestra en la imagen 3.2. En caso de no reconocer la instrucción, algún paso ha ido mal y se debe volver atrás.

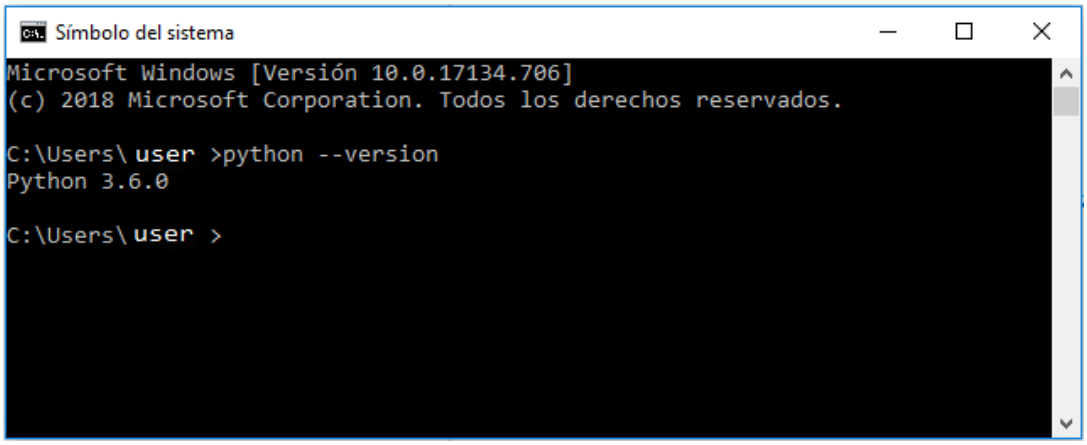

Imagen 3.2

Esta misma comprobación servirá para revisar la versión del pip instalada, bastará con introducir el comando `pip –version`.

PASO II Instalar Entornos Virtuales

Una vez instalado correctamente el entorno de desarrollo de Python, se instalará el módulo para crear entornos virtuales y se creará un entorno virtual donde finalmente se instalará TensorFlow. Para realizar esto se debe introducir el comando: `pip install -U pip virtualenv`.

A continuación se creará un entorno asignando la ruta que se desee con el siguiente comando de consola: `virtualenv --system-site-packages -p python ./venv`. Y trans crearlo se activará con el siguiente comando: `.\venv\Scripts\activate`.

Llegado a este punto, en el sistema se dispone de un entorno virtual activo útil para el uso de TensoFlow. Para comprobar su correcta creación se podrá hacer uso del comando: `pip list` que listará todos los componentes del pip. Cuando se desee desactivar este entorno bastará con usar el comando: `deactivate`.

PASO III Instalar TensorFlow

Finalmente, con todo el entorno instalado y activo, se podrá proceder a la instalación de TensorFlow. Para ello, bastará con usar el comando: `pip install --upgrade tensorflow` y se comprobará la correcta instalación con el siguiente comando: `python -c "import tensorflow as tf; tf.enable_eager_execution(); print(tf.reduce_sum(tf.random_normal([1000, 1000])))"`. Este paso es el más corto pero el crucial para el correcto funcionamiento de la aplicación.

Para finalizar con este apartado se confirmará que la versión del protobuf sea 3.6.0 con el comando: `pip list`. De no serlo se deberá cambiar la versión. Por suerte es tan simple como ejecutar el comando: `pip uninstall protobuf` para desinstalar el actual protobuf y seguidamente instalar la versión correcta con: `pip install protobuf==3.6.0` concluyendo así con la instalación de TensorFlow.

Para más información se puede acceder al siguiente enlace con una guía paso a paso de la instalación de TensorFlow con Python: https://www.tensorflow.org/install/pip

12. Glosario

A continuación, se resumen algunos acrónimos usados durante el desarrollo de esta memoria:

- CNTK: Microsoft Cognitive Toolkit
- API: Interfaz de programación de aplicaciones (en inglés, Application Programming Interface)
- UCM: Universidad Complutense de Madrid